UNIVERSITY COLLEGE LONDON

LEARNING FROM EXPERIENCE IN THE ENGINEERING OF NON-ORTHOGONAL ARCHITECTURAL SURFACES:

A COMPUTATIONAL DESIGN SYSTEM

Katrin Jonas

DECLARATION OF AUTHORSHIP

I, Katrin Jonas, declare that this thesis titled 'Learning from experience in the engineering of non-orthogonal architectural surfaces: A computational design system' and the work presented in it are my own. I confirm that

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, it has been clearly stated.
- Where I have consulted the published work of others, it is always clearly attributed.
- Where I have quoted from the work of others, the source is always given.
- With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself, jointly with others, I have made clear what I have contributed myself.

ABSTRACT

This research paints a comprehensive picture of the current state of the conception and engineering of non-orthogonal architectural surfaces. The present paradigm in the design and engineering of these elaborate building structures is such that the overall form is decided first and it is then broken down into building components (façade cladding, or structural or shell elements) retrospectively. Subsequently, there is a division between the creation of the design and then the reverse engineering of it. In most of these projects, the discretisation of elaborate architectural surfaces into building components has little to do with how the form has been created, and the logic of the global form and its local subdivision are not of the same order.

Experience gained through project work in the sponsoring company Buro Happold has been harnessed to inform the implementation of a design tool prototype. It is an open, extendable system. The development of the tool aims at stepping outside the current paradigm in practice; provides an integrated process of bottom-up generation of form and top-down search and optimisation, using an evolutionary method. The assertion of this thesis is that non-orthogonal design, which mimics a natural form in appearance, can be derived using mechanisms found in nature. These mechanisms, e.g. growth and evolution, can be transferred in such a way that they integrate aspects of the aesthetic, manufacturing, construction or performance. Designs are then created with an inherent logic. Growing form by adding discrete local geometries to produce larger componential surfaces ensures that the local parts and the global geometry are coherent and of the same kind.

The aspiration is to make use of computational methods to contribute to the design and buildability of non-orthogonal architectural surfaces, and to further the discussion, development and application of digital design tools in practice.

ACKNOWLEDGEMENTS

The author would like to acknowledge the support of the following people and organisations.

I would like to thank both of my supervisors, Prof. Alan Penn and Dr Paul Shepherd. Alan always contributed (more) smart ideas, and my discussions with him helped to achieve the balancing act between industry and academia. Paul knows that without him this would not have happened. Thank you!

I am grateful to the sponsoring company Buro Happold Ltd., especially Dr Mike Cook and Wolf Mangelsdorf, for trusting that someday the work would generate some fees. I am grateful for the freedom and support that they have given to the project. I would like to thank Emiliano Cevallos, Ying Chen, Lawrence Friesen, Alison Killing and in particular Jalal El Ali for being a wonderful team of exceptional people to work with.

Tom Makin was the geek on the project, and Dr Igor Siveroni, the true computer scientist; both contributed significantly to the ideas and the implementation of the tool prototype.

Aaron Caffrey taught me most of what I know about C# and we had long discussions on how different the operators for the genetic algorithm (GA) had to be. David Rutten gave the first ever C# for Rhinoceros (Rhino) workshop at Buro Happold and implemented the first version of my approximation idea as an example project. Martin Manegold converted the approximation plugin into a Grasshopper component. Michael Lilley and I worked on the growth model experiments, trying to grow closed componential surfaces in a naïve way.

Prof. Keith Ball provided some of the first ideas on how to tackle the combinatorics by programming small pieces of code, and is one of the giants of this study. I would also like to extend my appreciation to Marty Doscher for being the second giant.

Dr Chris Williams always had good advice for me when I passed by his office at Bath University. Prof. Ruth Conroy Dalton reminded me not to write a business plan but a thesis, and supported me at a point when I had good reasons to give up.

I would like to thank my former supervisors Prof. Mike Weinstock, Prof. Michael Hensel and Prof. Achim Menges, as well as my past collaborators Dr Martin Hemberg and Matthias Michel. They encouraged the concepts and the work which form the basis for this study.

I would also like to thank my colleagues Karen Martin, Erica Calogero and Chris Leung for continuous discussions and support. And I am particularly grateful to my friends, my sister Britta and my mother Wilhelma for bearing with me.

This research was funded by the Engineering and Physical Sciences Research Board (EPSRC) and Buro Happold Ltd.

The work is dedicated to my family.

THESIS STRUCTURE

This thesis consists of eleven chapters. The core of the document is in two parts, the Review and the Design Project. The Review encompasses three chapters and the Design implementation, six chapters.

Introduction (Chapter 1)

The Introduction gives an overview of the problem area that the research aims to address. The study investigates both the theoretical notions relevant to the field of design system development and their analogies to nature - in particular, growth, emergence and evolution.

PART 1 Review (Chapters 2 to 4)

The Review covers state-of-the-art research and practice. It looks at generative design processes. This is followed by an appraisal of the current realm in industry, with respect to the conception and engineering of elaborate building envelopes. The examination of the industry is supported by case studies that were live projects the author worked on at the sponsoring company during the time of the study.

PART 2 Design implementation (Chapters 5 to 10)

Design Implementation describes the development of a design system prototype, which is informed by both theoretical concepts and their practical application to the creation of architectural componential surfaces built from discrete geometry sets. The documentation of Design Implementation consists of six chapters. The first chapter, Chapter 5, is 'Growth Model Development'. It presents the experiments carried out to derive the Unit Cell growth model, which is described in Chapter 6. 'Surface Approximation', Chapter 7, is a validation study in which the principle of the Unit Cell concept is applied to a predefined surface. Chapter 8 depicts the implementation of the bottom-up growth mechanism to generate surfaces. Chapter 9 is Search and Optimisation. It comprises the embedding of the bottom-up surface generation mechanism into a search and optimisation engine, by employing a genetic algorithm. The last section of this part, Chapter 10, presents the application of the developed ideas. It includes one competition design which employs the Unit Cell principle and a workshop in which participants beta tested the implemented software tools described in the previous sections. The chapter also refers to potential future applications.

Conclusion (Chapter 11)

The Conclusion summarises the insights gained through the development of the prototype design system with respect to the prospects of the thesis, which are set out in the Introduction and refined through the Review. It records the achievements made, as well as suggestions for changes and further developments.

CONTENT

FIGURES

CHAPTER | 1

INTRODUCTION

1

CHAPTER | 1

INTRODUCTION

"(…) the idea of organic form is seen not so much in terms of static balance, but more as something which grows and develops out of the material. The form is integral or 'innate' to the work, rather than being preconceived and 'impressed' onto it." (Steadman 2008, p. 9)

Steadman, in his writings on Coleridge's view of Shakespeare's work, indicates a biological analogy in the arts that has been present since Aristotle. It suggests that beauty comes from coherence and balance in the relationship between the whole and its parts, both in its appearance and in its response to functional requirements.

Two fundamental changes in the arena of architectural design have occupied the author's interest: the distance to a preconception of form and the appreciation of multiple possible solutions to a single given description of a design problem.

Technology, and the ease in accessing it, has not just allowed for a change in the appearance of architectural design from orthogonal to non-orthogonal, but has also opened up the possibility of a different approach to the process of design itself.

The centralised approach to design, which starts with ideas of the final solution, e.g. the global form of a building, is supplemented by a decentralised approach using a design method that depicts the setting up of a framework describing the design problem. The solution is then developed through a process, rather than being preconceived. The potential, but also the challenge, lies in the definition of the regulators of the system that inform the design during its formation.

This study investigates the theoretical notions relevant in the field of design system development, in particular the adaptation of natural processes such as form growth and evolution. The study also looks at the current realm in the industry, with respect to the conception and engineering of non-orthogonal architectural surface designs.

The notion of architectural surfaces here comprises dividing structures, façades, roofs or envelopes which refer to the complete hull of a building. Envelopes often define an articulated surface that creates the roof, wall and support areas as one continuous structure.

1.1. Aims and objectives

The core objective is to develop and test an evolutionary design process, in the form of a user-interactive design tool prototype, for the creation of non-orthogonal architectural surfaces to proof-of-principle stage. The aim is the combination of bottom-up initiation of form with top-down search and optimisation, with the emphasis on informing the parameters by real-world considerations. The aspiration is to make use of computational methodology to advance the design and buildability of architectural surfaces, and advocate the application of digital design tools in practice.

The aim is to gain insights into the design and engineering of elaborate architectural surfaces, and in particular to study the relationship between the global form of a surface and the local geometric entities from which it is finally built. It is intended

that the recognition and extraction of recurring problems in the realisation process of elaborate surface design, be converted into suitable considerations for the tool's development. The intention of the user-interactive tool is to combine two mechanisms. These two mechanisms, or engines, are form growth and evolutionary search. The aspiration for the generative growth engine is to provide a discrete representation (growth model) of buildable local parts that come together to form global surface structures, combined with a parametric control of their repetition. The challenge of developing the growth model is to overcome the hurdle associated with creating continuous surfaces from discrete elements in a bottom-up approach. The aspiration for the evolutionary optimisation mechanism is the adaptation of search operators to the growth model and suitable fitness criteria.

The project aims to explore the notions of emergence, growth and artificial evolution. With regard to emergence, complexity is mostly derived from simple parts and simple rules of local interaction. In a growth process, geometry is created in an additive way, so that discrete parts add up to form larger entities. Artificial evolution, here, means that populations of forms are improved over the course of generations towards user-defined objectives. The research seeks to demonstrate how technology is used to step outside a dominant realm in industry. The norm is that an intricate form is designed without the regulating factors that it will later have to be subjected to through reverse engineering.

Exploring this problem in live practice, and thereby being able to extract key parameters for the prototype tool, represents a key contribution of this thesis.

The proposed development of an evolutionary design prototype requires three major mechanisms:

- form generation
- form analysis
- form optimisation or adaptation to variable criteria.

The associated objectives are:

- to develop a suitable growth model
- to establish fitness criteria
- to define suitable search operators.

1.2. Thesis statement

The thesis of this research proposes that the use of computational design methodology, and particularly bottom-up form generation, can side-step recurring problems present in the current dominant realm in industry. This realm is that non-orthogonal architectural surfaces are generally reverse-engineered. The growth model in a form-synthesising process can be informed by available or potential building components. The parameters, as well as the fitness criteria of such a tool, can control spatial configurations and performance through geometric measures. Thereby, a method is created in which an architectural model can adapt to changing requirements and

different considerations can be negotiated within the same design process. The idea is that the method can be defined on the basis of functional criteria through explicitly implemented parameters and fitness measures, and non-functional criteria through user control. Such a tool allows the exploration of possible solutions within a controlled geometric framework, which could not have been predicted otherwise.

1.3. Generative design processes

The presence of design methods is made possible by the available computational resources. The understanding of its potential has slowly found its way into architecture. Paul Coates describes this potential by saying that "…the computer can make a difference in the process of learning, when the computer is used not only as a drafting or modelling tool, but as a 'tool to think with'…" (Coates 2010, p. 26). "This approach seems to provide a nice paradigm for architecture as the emergent outcome of a whole lot of interconnected feedback loops, which replace top-down geometry and the reductionist tradition, with dynamic relations and emergent outcomes not defined in the underlying model" (Coates 2010, p. 1). These statements are based on the assumption that architectural design is or can be defined as a problem of parameter combinations, and that it can be solved through a non-linear route, in which multiple interrelated factors can be communicated. The designer authors and/or operates the regulators of a system, and, this way, only indirectly defines the outcome.

1.3.1. Craft evolution

Christopher Alexander, in his renowned book 'Notes on the Synthesis of Form', supported the distance to the design outcome through a systematic approach. He said that: "…the adaptation must take place independently within independent subsystems of variables" (Alexander 1964, p. 73). This suggestion was preceded by his reasoning that the contemporary dilemma in the arena of design is that the decentralised organisation of making through craftsmen has been lost. He argues that the individual in vernacular production acted as an unselfconscious agent, following the rules of established traditions. Changes to the process of making were only applied as reactions to local problems, thus triggered by functional necessity. The adaptation of the rules of making to the product to be achieved happened over long periods of time. The problem today, he reasoned, is that the "unselfconscious" craftsman has been replaced by self-conscious designers, who are confronted with ever-changing requirements and expectations (Alexander 1964). Steadman summarises Alexander by saying that unselfconscious collective making has been replaced by "intuitive individualism" (Steadman 2008, p. 163).

Alexander proposes that design methods are needed to respond to the complexity of contemporary design questions, forming a way out of the dilemma. Steadman summarises Alexander's prescription as "…a new type of mathematical or systematic design method which will produce results equivalent to craft evolution by simulating its mechanism" (Steadman 2008, p. 172). While Alexander does not explicitly use the notion 'craft evolution', Steadman argues that the organisation, which from Alexander's point of view produced well-adapted designs, described the mechanisms of evolution. The author agrees with Steadman, who suggests that the

comprehensive reasoning of Alexander's argument guides us to the suggestion that what used to be achieved over long periods of time, the passing on and updating of know-how, can be attempted through the use of computational design methods. This leads to the interesting proposition of using computers as a means to mimic traditional craft evolution.

1.3.2. Analogies to nature

This thought lead to Steadman's analogies of design to nature (Steadman 2008). He distinguishes between different categories and the design implementation in this research acknowledges two of them: 'growth' and 'evolution'.

'Growth' here refers to the bottom-up accretion of smaller geometric entities to form larger structures: architectural surfaces. 'Evolution' in this context is defined as computational routine, in which architectural surfaces are improved over the course of generations.

The development towards non-orthogonal designs in architecture suggests a comparison to nature, where non-orthogonal form originates. There has also been an extension of interest from the creation of form which looks organic, to the study and employment of theories which mimic biological mechanisms, both in the way form comes together and how it adapts.

1.3.3. Computational evolution in architecture

Computers allow the compression of time-based processes, such as evolution. 'Artificial evolution' is an established computational method, where trial and variation guide the adaptation to specified criteria.

Artificial evolution is widely used in disciplines such as biochemistry, e.g. to simulate the growth of viruses. In finance, multiple generation growth patterns are mimicked to promote adaptation, in order to predict the stock market. In engineering, the method is employed to optimise specific features in designs. Artificial evolution is used on non-linear problems, where the solution to a problem cannot be predicted. This research employs the mechanisms of evolution to explore buildable form.

1.3.4. Optimisation versus exploration – the definition of objective functions

Design methodology originated in engineering, where clearly defined functional problems are solved, and it is an established field in both research and industry.

Paul Coates (Coates 2010) recognises that design methods were used to increase the capabilities of a product, while trying to reduce the cost. The problem can be complex: "…the fundamental idea", Coates writes, "remains that an optimum can be defined. This has always been a problem for architecture, where the incommensurability of many classes of functional criteria prevents easy optimisation" (Coates 2010, p. 160).

The transfer of the concept of design methodology to architecture shifts the emphasis from pure problem solving to exploration, and the question becomes how to first define and then implement relevant objectives.

Steadman (Steadman 2008, p. 2) sees a potential to extend functional criteria in design methods to include architectural features. As functional considerations, he describes building materials, building elements, the engineering of structures and the performance of all of them with respect to environmental factors such as heat, light and sound. To also embrace architectural features Steadman suggests, for example, the inclusion of the geometrical organisation of their parts and structures into design methods.

The author agrees with Steadman and would add that user interaction in design tools seems to offer a broad means to complement explicit functional criteria, for which there is a known optimum, with soft objectives around experience and aesthetic preference. Work has been done at both ends of the spectrum: pure optimisation and pure formalistic exploration. There seems to remain an interesting middle ground to be explored, which this research has investigated.

1.4. Problem domain – leitmotif

While many people on whose work and support this study is based could be named, there are two men whose observations became the guiding themes for this research. They can be summarised in the following two sentences:

> "This is to place one order on top of another one." (Doscher 2006)

> "One can't grow form in a naïve way." (Ball 2007)

The conversations the author had with Marty Doscher and Prof. Keith Ball are recalled and documented here.

1.4.1. Placing one order on top of the other

One important discussion for the author occurred at the beginning of her studies in 2006 during a SmartGeometry workshop on parametric modelling in Cambridge. The supervisor was Marty Doscher, a senior designer at Morphosis Architects. The author attended the workshop bringing along a live project, the Glasgow Museum of Transport, designed by Zaha Hadid Architects and engineered by Buro Happold. The problem of tiling the complex architectural geometry with a regular tiling pattern on the façade was investigated. At that time, the design of the façade was to attain a homogenous tessellation all over the outer hull, of similarly sized, rectangular steel panels, laid out in a fish-scale pattern.

A geometrically coherent solution would have been to scale the size of the tiling with the size and curvature of the building, but that was not wanted, first, for aesthetic reasons, and second, because it was assumed that it was more economic to use the same size of panels, which only needed to be cut and bent into shape. This assumption is a common falsity, as will be seen later. When looking at the problem, Doscher (Doscher 2006) stated that it is a common phenomenon when an attempt is made to place one order on top of a completely different one. What he described in a single sentence was that the global geometry of the building had a completely different organisation to the local geometry of the tiling pattern. He then spoke about the San Francisco Federal Building, a design by Morphosis under construction at the time, and described its perforated metal sunscreen. Some of the fold lines of the façade run

diagonally. The pattern on the metal was consciously kept orthogonal, resembling an ordinary brick layout. At the fold edges of the metal layer, the brick pattern just ran over the diagonal fold so that it was visually cut diagonally (Figure 1). Doscher said that it was a design decision; they wanted to play with the theme to place two orders on top of each other. The beauty in their design lies in that the decision was taken consciously. The project comments on contemporary design at a point in time where the handling of dramatic form has yet to be explored. In most intricate architectural designs, the overlay of different logics has almost always undesired effects to do with detailing errors and cost. Raising awareness of the relationship between the local and global geometry became a fundamental part of the author's project work, which is documented in the case studies of this thesis. This understanding supported the proposal for a bottom-up generation of form with discrete parts, which only ever produces coherence between the local and global organisations.

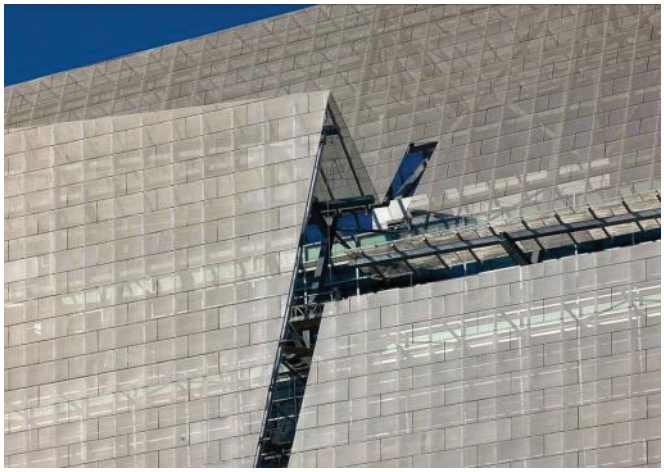


Figure 1
San Francisco Federal Building by Morphosis Architects. Façade section.

1.4.2. 'Growing form the naïve way'

Another important conversation was with Prof. Keith Ball from the Department of Mathematics at University College London (UCL). It took place during the process of developing the growth model for the tool prototype. Among a few others, he reacted to a call that was sent out to different Departments of Mathematics in the United Kingdom. It asked if they were familiar with any work that had been conducted on repetition in three-dimensional geometry meshing or, more generally, the relationship between the discrete parts of a mesh to the global geometry. Ball appreciated that the dependency between the local component and the catalogue of global configurations had been recognised. He stated that form could not be grown with a naïve approach (Ball 2007). His statement suggested that it was tedious to try growing continuous configurations with made-up geometric rules, by going through all the possible combinations of shapes and angles. The author understands that Ball means it is impossible to establish what all the geometric rules are using trial and error. In this conversation, Ball encouraged the author to continue the experiments in which the growth model was related to an underlying matrix. He inspired the tile factory of the tool prototype through small pieces of code that he produced, which set the author on a path to define the combinatorial procedure in the Unit Cell. The assumption that could be made after this conversation was that in form generative processes,

the L-system is often an underlying principle, and shapes are branching objects or staggered building blocks. However, what seemed little explored were the definition of a geometry system or grammar that allowed the generation of continuous surface geometries, and a control of the types and number of elements.

Ball's own investigations led him to develop a program for dECOi and Foster + Partners for the façade development of the Swiss Re headquarters in London, better known as 'The Gherkin'. The program showed the catalogue of possible global configurations that could be derived using a diamond-shaped façade element. If the parameters of the shape and its connection angle to its neighbours changed, the global geometry changed too. Ball talked about the difficulty of explaining the fact that a restricted local geometry can only create a restricted, and not an infinite, catalogue of global geometries. Wanting to place a particular local geometry, such as the diamond shape, on to a global form that is not one of those within the form catalogue of the diamond does not work. Either the architects' idea of the local shape or their expectations of the global form would have to change. Ball can now see 'The Gherkin', which has local and global geometries of the same order, when he looks out of his office window at UCL.

Both conversations shed light on the two main problems that the author was facing, both in the project work at Buro Happold and in the development of the design method. On the one hand were the discretisation tasks in the reverse engineering of elaborate architectural surface proposals and on the other hand the attempt to establish a growth model for a bottom-up form generation process.

1.5. Research context

This research was sponsored by Buro Happold, an international building engineering consultancy. The author was a member of the Generative Geometry group (GG group) during its existence from 2006–2010, which was part of Building Structures. The core members of the GG group came from an architectural background, but it also hosted secondees from engineering, mathematics, computation and various other disciplines. The work provided an interface between architects and engineers. The GG group influenced designs quite boldly, possibly because of the design background of its members, combined with an interest and understanding of engineering considerations. The group conducted early experiments to produce design proposals, by what one might say was a 'misuse' of performance analysis (structural, wind and sunlight simulation), then reversing the performance impact. Informing design by performance factors is now a more common approach in practice than it was back then. At the beginning of the author's involvement in the project, which coincided with the establishment of the group, every project brief was a pilot study and each solution was developed from scratch. Over time, problems reoccurred, and this way, solution methods could be reused in different projects. The work comprised the understanding and solving of geometric problems, mostly to do with defining a structural model in correspondence to an intricate form or to tessellate elaborate design. It also included the setting up of automated procedures for detailing, design conception, comparative analysis and optimisation, and the development of alternative designs and structural solutions.

Buro Happold has gained a reputation for encouraging the creation of computational processes. This is rooted in a collaboration with Frei Otto; together with the founder of Buro Happold, Ted Happold, they advanced the process of 'form-finding' for Frei Otto's projects. Designs evolved as minimal surfaces between boundaries by using physical models: soap films or stockings were suspended between bounding frames. At a later stage, the physical simulation to derive surfaces with a constant mean curvature was translated into a digital simulation application, in-house software called 'Tensyl'. The tool allows the study of membrane structures through the simulation and analysis of minimal surfaces. The development of Tensyl was followed by other computational process developments, mostly to rationalise the tessellation of elaborate surface designs. One tool will be introduced in the Industrial Review section later in this document. This research continues the investigation within Buro Happold by developing and deploying digital processes.

1.6. Methodology

The core research objective is met in four distinct stages, each associated with a different method. These stages are harvesting opportunities, experimenting, implementing and testing. The review intends to harvest opportunities. The experimentation and implementation are concerned with the actual development of the user-interactive tool prototype. The final testing comprises firstly the test of the operational performance of the design implementation and secondly the application of the growth model principle in a project for a competition.

Review

First a review is conducted that is separated into three parts. The first part is the examination of existing design processes that combine the generation and search of architectural form. From this examination, it is possible to determine what opportunities there are to extend the field.

The second part of the review looks at the current industry in relation to designing and engineering non-orthogonal architectural surfaces.

The appraisal of industry is supported through the third part of the review, the case studies, that were carried out at the sponsoring company.

Particular attention in the review of industry and the case studies is placed on the relationship between the global form of a surface and the local geometric entities from which it is built.

Parts two and three of the review allow for the recognition and extraction of recurring problems in the realisation process of elaborate surface designs, which are then converted into suitable considerations for development of the tool. This part of the review also intends to outline the opportunity for the application of a design tool in practice.

Experimentation

Second is the experimentation that focuses around the development of the growth model for the form-generating engine of the tool. The experiment investigates what the local geometries have to look like to constitute a global articulated form.

The testing looks at ways to define discrete local parts and rules from which various continuous, articulated surfaces could be generated. The work leads to the development of a principle that allows the generation of geometry sets. The principle is tested by approximating the given surface geometries in a top-down approach, before being implemented in the bottom-up generation process.

Implementation

Third is the implementation that seeks to find a sensible, useful and stable architecture for the program to support the user-interactive tool. The tool has to combine the inductive growth of form with the evolutionary technique for search and optimisation. This implies the setting up of a computational definition of form growth using the developed growth model and tailoring of the operators of a classic genetic algorithm (GA). The growth model principle is implemented into the GA, including three- and four-sided shapes. The program architecture is set up to allow the extension of further shapes in the future.

Test

The fourth and final stage is twofold. The first part is the testing of the program to ensure that the code is stable and the optimisation works. The second test is the application of the growth model principle in a design competition that allows the gathering of initial feedback on the usability of the geometry system and provides a view of sensible functionalities for the process.

PART 1

REVIEW

The review of this thesis is structured in three sections. The first gives an overview of the generative design processes relevant to this study, which are mainly found in research. The second section covers the current state of the industry, looking at the conception and engineering of elaborate building envelopes. This is supported by the case studies in the third section of the review. These are live projects the author worked on at the sponsoring company during the time of this study.

CHAPTER | 2

DESIGN PROCESSES AND EVOLUTIONARY SEARCH

2

CHAPTER | 2

DESIGN PROCESSES AND EVOLUTIONARY SEARCH

A computational design process, or a design tool, is a digital framework for design generation and exploration. It is a system of parameters in which, similar to a switchboard, different factor weightings and their combinations can be explored. It is employed where the outcome of parameter combinations, even when there are a small number of them, cannot be predicted.

Design tools in this context are not simply altering geometric configurations, as in a parametric model, but instigate the coming together of form.

The catalogue of solutions associated with every design process is finite. The range within this catalogue can be small or very large. If the pool of solutions is extensive, than the design generation is generally supplemented through a search mechanism. Search can imply not just the finding of good solutions but also their optimisation. Solutions which are found to be good are then improved by applying specific optimisation operators.

Design methods originate in the field of engineering where explicit functional requirements are needed. In their application in design, their parameters can be extended to include aesthetic qualities, as well as considerations relevant to the manufacture and construction of the designs or to their performances. Design processes, and in particular evolutionary techniques in architecture and engineering, are often used either for explicit technical optimisation or formalistic form generation. Only a few examples consciously investigate the middle ground.

This review introduces selected design methods which are relevant because they combine the generation of form with search, and because one or more of the following criteria apply:

- the combination of inductive generation of form and top-down search
- the inclusion of user interactivity, thereby the consideration of user preference
- the provision of a discrete geometric representation
- the creation of buildable form
- the implication of functional or performance measures relevant to architecture
- the use of artificial evolution
- the design subject in question is the architectural surface.

The table in Figure 2 lists the selected design methods and indicates aspects that their implementation considers.

	Tool	Generation	Search	Optimisation	AE*	Applied	RWP**
Groningen Twister	●	●	●			●	●
LEGO structures		●	●	●	●		●
Truss design generation	●	●	●	●	●	●	●
EifForm	●	●	●	●		●	●
Genr8	●	●	●	●	●		
ECSS	●	●	●	●	●		●

*AE Artificial Evolution RWP** Real-World Parameters

Figure 2
Table of presented design processes. The table shows which aspects are implemented in each process.

2.1. Design tools – generation and search

Design tools are user control processes. The user can author the framework within which designs evolve. He or she can also decide against which fitness criteria the solutions are measured. By default, these include user preference. Depending on the architecture of the system-specific parameters, fitness criteria remain open for analogue testing. Often, there is fitness weighting as well. Such a tool allows the exploration of a problem domain rather than solving a specific problem. The advantage of user control is that soft measures such as experience and preference are included, along with analytical and quantitative evaluations.

The development of software design tools based on the principles of complex systems was pursued by the Emergent Design Group (EDG) at MIT (Testa et al. 1997). Between 1997 and 2001, the group produced a number of different tools, 'Genr8' (Figure 8), 'Weaver', 'Agency', 'MoSS' and 'GermZ' (Figure 3). These tools emphasise the interaction with the user and the idea of presenting multiple solutions to a given design task. The tools created by the EDG provide visually compelling output because they employ advanced growth algorithms. The geometries generated by the tools carry no direct physical information and solutions need to go through long post-processes in order to be translated into the physical realm. So far they have mainly produced graphical designs and small scale models. Genr8 (Hemberg et al. 2008) was widely used in architectural education. It will be discussed in detail later in this chapter. Akin to the other EDG tools, Genr8 has not been applied in architectural practice due to its lack of physical considerations. The user faces a significant challenge trying to convert physical aspects into the available parameters (Hemberg 2001).

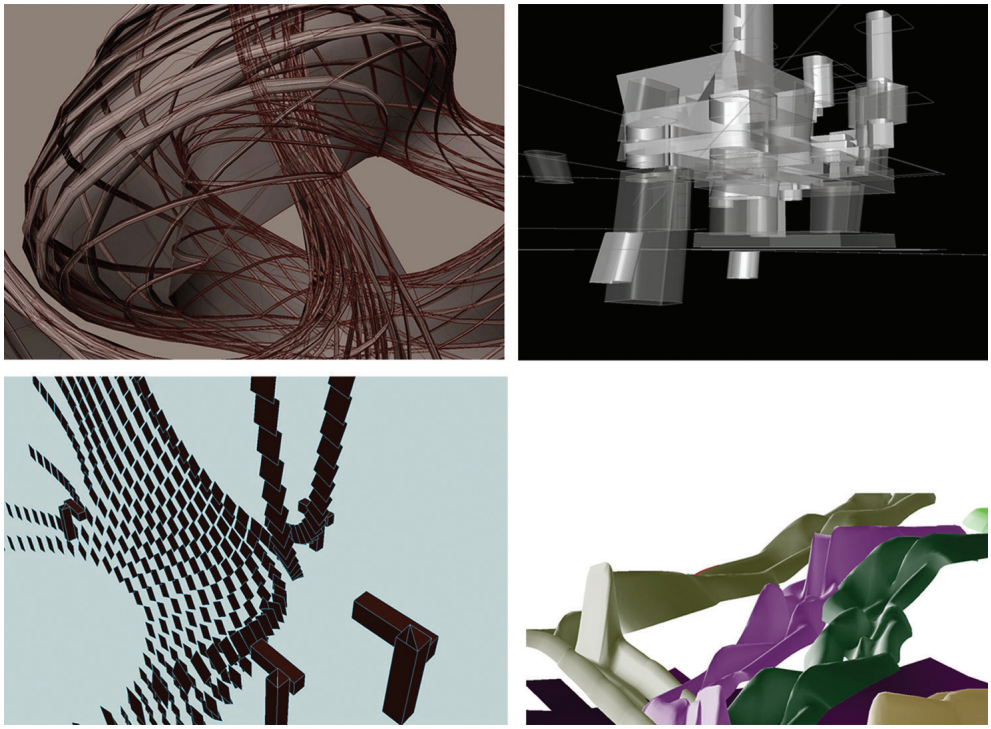


Figure 3
EDG tools (clockwise from top left): 'Weaver', 'Agency', 'MoSS' and 'GermZ' (EDG, MIT, 1997).

Groningen Twister

An interesting tool which was created for application in a live project is the Groningen Twister (Scheurer 2003). The method arranges different types of columns in an underground bike store using a self-organizing clustering algorithm (Figure 4). The placement of the columns is influenced by the structural requirements, as well as the functional aspects of the space. The system produced only one outcome. There is also no comparative mechanism for how good a solution is in comparison to other options, and there is no mechanism for optimisation involved.

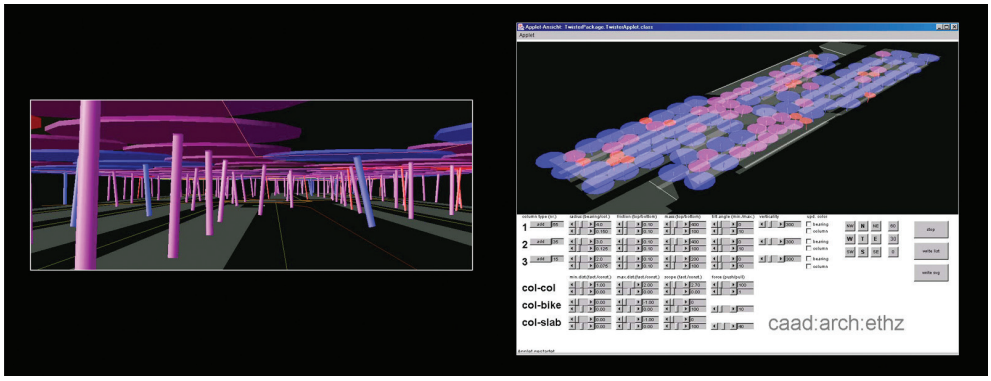


Figure 4
Groningen Twister. Screenshot viewing one solution for the distribution of column types from within the underground bike store (left), and isometric overview of the same solution (right) showing the whole site and the user interface (Fabian Scheurer, 2003).

EifForm

EifForm is a computational design and optimisation tool that combines the generation of buildable 3D truss structures with geometric analysis and optimisation. It was developed by Dr Kristina Shea, Marina Gourtovaia and Dr Xianzhong

Zhao at Cambridge in 2000. The process combines structural grammar (SG) and performance evaluation into an algorithmic technique called structural topology and shape annealing (STSA). The user can define geometric constraints such as spatial boundaries (maximum height, boundary lines and internal headroom), maximum and minimum lengths of members, minimum angle between members and maximum number of members meeting in one node. No solution may be found if too many constraints are applied. One built application of the tool is the project named 'Hylomorphic' (Figure 5). It was initiated by the architecture practice OSA for the exhibition 'Gen[H]ome Project: Genetics and Domesticity' which it organised at the MAC Centre for Arts and Architecture in Los Angeles in 2006, and which was developed together with Arup Engineers and Kristina Shea (Shea 2003). The solution structure was developed from a planar grid that pushed itself up, iterating through different node positions. The constraints here were the spatial boundary and the maintenance of the triangulated grid.

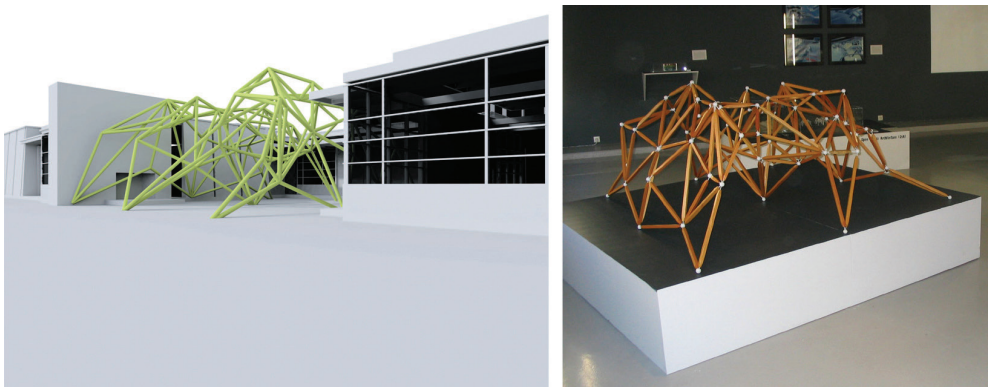


Figure 5
Hylomorphic. Rendering (left) and model (right) of the Hylomorphic project (OSA, 2006).

2.2. Evolutionary design processes

Evolutionary techniques in architecture and its engineering are often used either for technical optimisation or formalistic form generation. An early pioneer in the field of creative form generation in architecture using evolutionary techniques is John Frazer, who started working on these ideas as early as the 1960s (Frazer 1995), advocating the concept of creative design tools (Janssen et al. 2002). In his experiments, evolutionary algorithms are used for facilitating form generation without any adaptation towards technical objectives. Paul Coates et al. investigated a combination of Lindenmayer systems with genetic programming, which allowed the production of recursively defined three-dimensional objects (Coates et al. 1999). The emphasis here, as well, was the exploration of form without any technical constraints.

LEGO brick structures - evolutionary process with structural performance criteria and discrete geometry

Phil Husbands, Giles Jermy, Malcolm McIlhagga and Robert Ives syndicated creative form exploration with the optimisation of structural performance in their experiments (Husbands et al. 1996). They investigated evolving three-dimensional LEGO brick structures using a discrete shape grammar and evaluated the outcomes according to their structural stability (Figure 6).

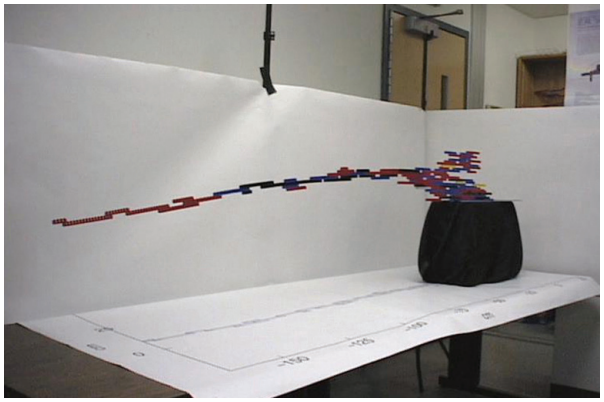


Figure 6
Evolving LEGO brick structures (Husband et al., 1996).

Truss design generation

Breanna Bailey and Anne Raich from Texas A&M University incorporate user design preference into a genetic algorithm which generates structurally sound large-span roof trusses (Bailey, Raich 2006). This way, they combine the soft objective of aesthetic preference with technical functionality. The optimum diagram for structural performance and material saving to a given span is known. By introducing user preference, the profiles vary and become visually more complex. The user choice encourages topological exploration while preserving good structural designs (Figure 7). The method is offered as a service to clients of a construction company in the United States.

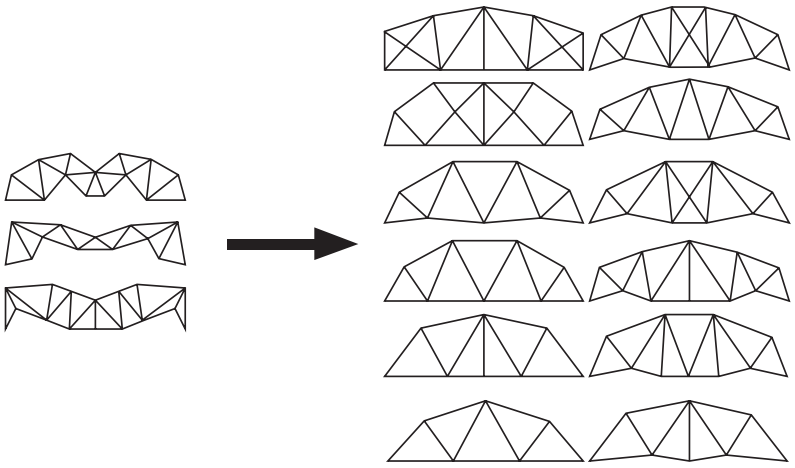


Figure 7
Parent and child generation of truss layouts (Texas A&M University, 2006).

2.3. Genr8 and ECSS – main reference projects

Important reference projects for this research are two existing surface design tools: Genr8 and ECSS. Both are interactive design tools for designers and architects to generate architectural surfaces, combining form generation with form optimisation. The aim of describing them in this research is to elaborate on the strong points of each project, while suggesting ways to overcome their weak aspects.

The outcome of the investigation of these two projects became the guideline for the design implementation presented in this thesis.

Genr8 was developed by Martin during his time at the Emergent Design Group (EDG) at MIT in 2001 (Hemberg 2001). It has been applied in graduate programs, e.g. at the Architectural Association in London, the Wentworth Institute of Technology in Boston and the University of Technology in Delft, as part of the course curriculum. Genr8 serves as an introduction to generative design processes in general and the use of artificial evolution in particular.

ECSS was a spin-off from Genr8, developed as a Masters dissertation project by the author of this thesis in 2004 (Jonas, Hemberg 2006). Both tools are hosted by the CAD application Maya, formerly an Alias Wavefront product that is now owned and developed by Autodesk. Genr8 is implemented as a C++ plug-in to Maya, while ECSS uses the Maya-embedded scripting language MEL. Both projects have not been used in industrial practice.

2.3.1. Genr8

Inspired by the growth process in nature, Genr8 uses shape grammars to grow architectural surfaces according to user-specified parameters and environmental conditions (Figure 8). The shape grammars are at the heart of the tool. The user can choose from three types of grammars: predefined, user-defined and evolved. The evolved grammar encompasses an evolutionary cycle for the development of shapes, according to the fitness criteria that the user inputs at the beginning of the process. This grammar is called the Hemberg Extended Map L-System (HEMLS), referring to Martin Hemberg who invented Genr8. HEMLS is an extension of the Lindenmayer system (L-system), a grammar developed by Astrid Lindenmayer to simulate plant growth. Apart from the choice of which grammar to employ, the user has four control interfaces: scaffolding parameters, environmental parameters, genetic operators and fitness criteria. Each interface has ample parameters and sub-parameters. The vast number of parameters is ambivalent. On the one hand, the variety allows the creation of many different and intriguing form results, and on the other hand, Genr8 has been criticised for offering too many parameters, especially those that can suppress one another when activated simultaneously. The user can feel like an observer of a random process, because it is difficult to recognise the effect that the input parameters have on the outcome. This is especially the case when default parameter settings are changed, and when none of them are un-ticked. Satisfactory results are achieved, in the sense that the reaction to the setting can be seen, when the user keeps to a strict testing sequence of changing only one parameter at a time over long and repeated generation runs. Another disadvantage is that the tool is not informed by real-world parameters. In particular, when the evolving grammar is used, local geometries overlay each other. The digital model has to be post-processed a lot before only a small-scale physical model can be produced, which is far from a full-scale prototype (Hemberg et al. 2008).

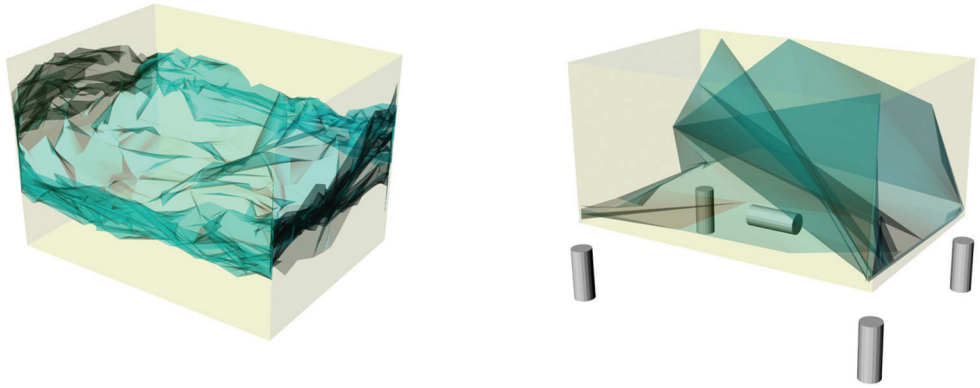


Figure 8
Genr8 surface examples. Two renderings of Genr8 surfaces are shown in the geometric environments to which they adapted. (Dr Carlos de la Barrera).

Referring to Steadman's categorisation, the analogy to nature in Genr8 is twofold. The principal method in Genr8 is the 'growth' of surfaces, which means that smaller geometries are configured to form larger geometrical entities in a bottom-up manner. The development of the surfaces is influenced by the environment that they are growing in. For example, the form avoids obstacles that the user draws in the graphic interface.

The second analogy is the use of an evolutionary technique to optimise the grammar according to the user fitness input. This, however, is only the case if the evolved grammar parameter is chosen. The process of selection and improvement takes place by employing evolutionary operators such as crossover and mutation. The problem with Genr8 is that users believe they are using an evolutionary process, when what is happening is that surfaces grow under environmental conditions (Figure 9).

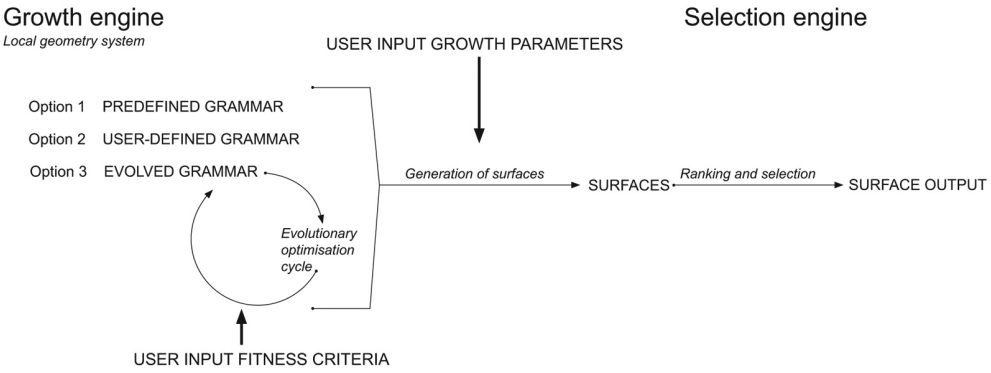


Figure 9
Program structure of Genr8.

2.3.2. ECSS

In contrast to Genr8, ECSS has very few parameters and a discrete number of components. The program builds upon the model of Genr8, in which surfaces are generated and optimised under user-defined parameters and fitness functions. In Genr8, the evolutionary optimisation takes place on the grammar level and only if the user chooses the option to evolve the grammar. The ECSS, in contrast, always runs the evolutionary cycle. Here, the optimisation takes place on the global surface level, not on the grammar level. The individuals, thus surfaces, of each generation are assessed

and the geometry that is found by the process to be closest to the optimal provides the genome for the next generation. Again, a set of individuals rather than a single offspring is the outcome. During this process of generation and selection, the user can change the environmental conditions and/or fitness criteria in which the surfaces grow. In this process, geometries that grow under the same conditions can exhibit exclusive forms, while maintaining shared features.

While Genr8 has a multitude of grammars, i.e. components from which the surface geometries are built up, ECSS stresses the idea of growing a multitude of surfaces from a very limited number of components. The system includes five hard-coded shapes and their respective rotations. The user can define which shapes are used in the process. Thus, if desired, only a subset of the five forms and their rotation can be employed. However, it is not possible to add any new forms (Figure 10).

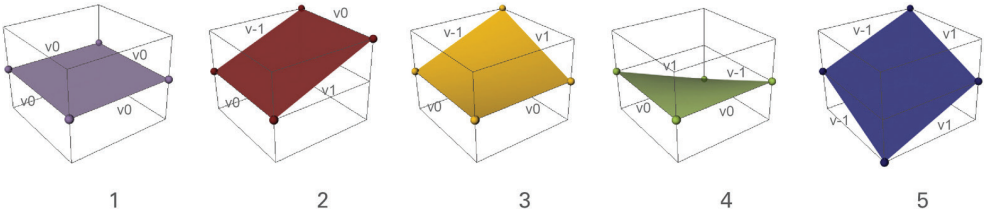


Figure 10
Hard-coded geometry set of the ECSS tool.

This was intended to study the emergent properties of the process. In this process, complexity can be built up using a small number of simple parts and simple rules that define how the parts are connected to form larger surfaces. The resulting surfaces are deductively controlled through a search and optimisation engine (Figure 11).

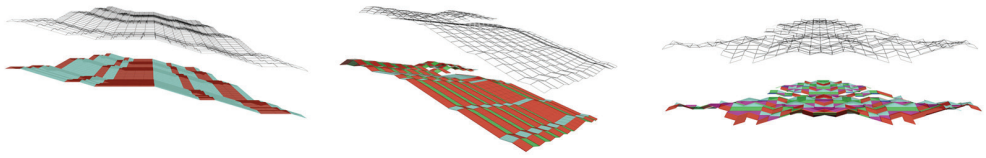


Figure 11
ECSS surfaces.

Unique to the system is that the components themselves, as well as the fitness criteria, represent considerations of the spatial, structural and manufacturing kind. Examples of the fitness criteria are the points of support where the structure reaches the ground, distances between the supports and the maximum height of the structure. The default settings for the parameters and the fitness criteria function as an engineering rule of thumb. They suggest the size of the individual component, according to the standard maximum transportable size of the building parts and the corresponding maximum spanning distance between the supports.

An additional fitness evaluation was provided by linking the Maya application with the finite element analysis application ANSYS. An automated loop allowed for comparative ranking of surfaces according to their structural performance, looking at deflection and stress.

2.3.3. Conclusion on main reference projects

The main strength of the ECSS is its closeness to reality; the fact that every surface solution can actually be built. Interpreting the idea of emergence into a simple set of repetitive parts, and simple rules for connecting them, implies an economic advantage. This is because up to this point in the building industry, for most types of structures, repetition means a reduction in cost and a likely decrease in the construction time.

The other benefit of the ECSS is that the reduced number of parameter settings allows the user to monitor the reaction to their input and makes the evolutionary process discernible.

The drawback of the ECSS tool is that the outcome surfaces are restricted to a specific appearance. Despite the multitude of possible configurations, every solution that comes out of the process is recognisable as being built up of all or a substitute of the provided elements. Surfaces created in Genr8 are far more differentiated due to the choice between three types of grammar: predefined, user-defined and evolved. Despite the use of the evolving grammar being unsatisfying, the fact that there is an optimisation cycle on the grammar level is an intriguing idea.

Aspects to bring forward to the research project of this study are the independent geometry engine of Genr8 and real-world considerations, such as the control of repetition in the gross model in ECSS (Figure 12).

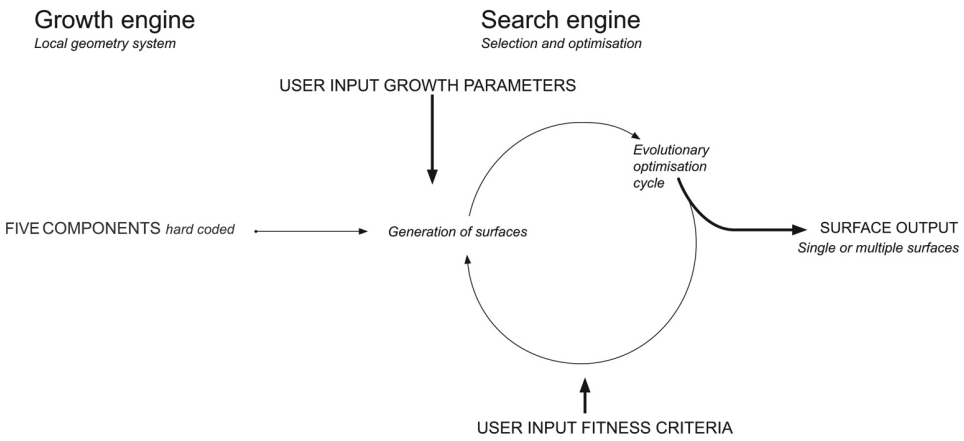


Figure 12
Program structure of the ECSS.

2.4. Summary

The review introduces design processes which showcase the combination of generation of form and the search for good solutions. All of the presented projects contribute to the field of decentralised generation of form relevant to architecture. The position between formalistic experimentation and targeted optimisation is an interesting middle ground of controlled exploration, which has not been exploited yet and is hardly applied in practice.

The authorship and/or supervision of parameters is becoming an important aspect of design processes. The projects in this review show that user interaction forms one opportunity to include soft objectives in design tools. In optimisation processes this

interaction allows the user to interpret the objective measures and sidestep suggested solutions, in order to make choices based on personal experience and preference.

An important aspect of tool development is the growth model that is used to create these forms. The growth model in the project that evolves LEGO brick structures, the truss design tool, the Groningen Twister and the eifForm utilize common building components and manufacturing techniques. This explains why these tools are used in practice or to provide physical one-to-one built examples. Their strength is that they explicitly represent the physical realm. This is not the case, for example, with the EDG tools. However, the latter are likely to produce more visually compelling designs.

The ECSS project proposes a kit of parts, which is not standard due to the double curvature in two of the five parts. However, the kit conforms to physical and economic constraints, as the components can be manufactured in a variety of materials. The growth model offers economic advantages, as the parts are repeated. This aspect is furthered in this study through the development of a growth model for the design implementation. The ECSS kit of parts forms the basis for the Unit Cell.

CHAPTER | 3

ENGINEERING OF NON-ORTHOGONAL ARCHITECTURAL SURFACES

3

CHAPTER | 3

ENGINEERING OF NON-ORTHOGONAL ARCHITECTURAL SURFACES

3.1. Introduction

Advances in digital technologies have led to rapid changes in architectural design, with a shift in aesthetics from the modern orthogonal building envelope to elaborate curved and folded forms.

Software that was initially developed for a different industry than architecture, such as Maya for animation and Rhinoceros for jewellery, broke down every geometric restriction on what building shapes could be conceived. These applications have now become a common tool set for architectural design, but the technical challenges that arise from non-orthogonal form designs are still being explored.

The subsequent dominant realm in the design and engineering of elaborate forms is such that the global geometry is provided first, before it is broken down into the necessary building components (façade cladding, structural elements or shell elements). There are techniques available to construct an elaborate form as a monolithic whole and efforts are being made to provide new ones. However, this review focuses on the situation in practice, where a subdivision of a global form is needed or where the definition of local geometry has been a conscious part of the design conception.

Often, there is a division between the creation of the form and its reverse engineering; the discretisation has little or nothing to do with how the design has been conceived in the first place. Sophisticated procedures are in place and continue to be developed to tackle this deductive task. The need for smart measures has led to a new type of service in the building industry, which concentrates on the computational detailing of non-orthogonal building hulls (designtoproduction; Evolute). The effort usually implies, not just breaking the structure into smaller parts, but also considering aspects of fabrication and construction to try to reduce cost. Even though these tools are being rapidly adopted there is still a substantial amount of time, and subsequently cost, going into reverse engineering designs this way.

The aim of this chapter is to make a distinction between the different types of non-orthogonal forms, and what this means to the relationship between their global geometry and the local components that the global geometry consists of. The objectives of reverse engineering non-orthogonal forms are introduced, followed by a short description of various reverse engineering methods. This section also presents an alternative approach, where the realm of top-down discretisation is turned on its head. This is by processes in which designs are created using a constraint geometric mode of modelling or are generated in a bottom-up way from discrete local geometries, e.g. material products or grander modular entities.

3.2. Non-orthogonal form – two distinct aesthetic approaches

A distinction can be made between two different interests in the aesthetics of elaborate form in architecture. The first one is that the global form is understood as a monolithic sculpture, where the detailing of the overall geometry is not part of the design. In fact, the aim is to overcome the need for subdivision. The second interest implies the assembly of global form through local parts, thus the detailing of a form is a conscious part of the design. This aesthetic division is not clearly reflected in existing manufacturing and construction industries. Technological developments are on their way to respond to a sculptural reading of architectural form. Here, one main field is the development of large-scale rapid prototyping, which, for instance, aims to plot the global form in one piece. An example is Enrico Dini's large-scale sand plotter (D-shape). An existing method in the construction industry would be to spray concrete onto a moulded mesh, as in the case of the cocoon-like Darwin Centre at the Natural History Museum in London. The visual seams here are expansion gaps milled into the initially continuous concrete shell retrospectively (Figure 13). However, in the majority of geometrically complex designs, it is necessary to subdivide them into smaller building components.

Figure 13
Darwin Centre. The cocoon-like Darwin Centre in the National History Museum in London was constructed as one continuous form by spraying concrete onto moulded mesh, and then the surface was sanded and polished on site.

The aesthetic approach that entails attention to the design of local components, provides the opportunity to incorporate fabrication and building rationale into the initial design conception. However, this opportunity is not always taken advantage of. There are numerous geometrically exciting proposals for componential designs that do not include physical constraints. Examples are designs which use a function in the parametric software application GenerativeComponents (Bentley Systems), which enables the proliferation of a designed component onto a master surface. The pitfall from the manufacturing point of view is that the multiplied component adapts to the referenced surface. While the visual features remain the same, the actual geometry of each component changes with the articulation of the surface with which it becomes associated (Figure 14). This does not play a role during the rapid prototyping of a small model, but it becomes relevant during the planning of a large-scale structure.

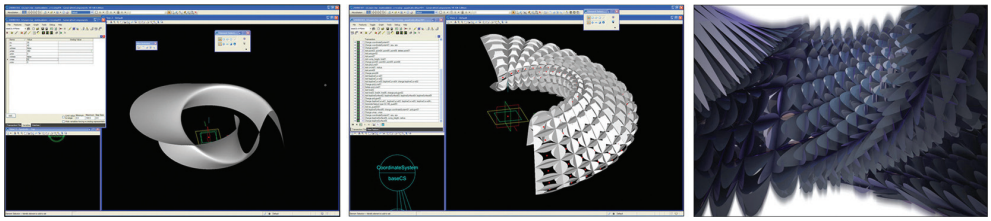


Figure 14
The figure shows a GC component that is proliferated over a master surface (Elif Erdine, 2008).

3.3. Complex forms, freeforms and hybrids

The author of this thesis will divide non-orthogonal architectural designs into three different categories. These are complex forms, freeforms and hybrids. Complex forms are those that despite their elaborate appearance are created from a rule-based framework that is reflected in their geometry, providing them with some kind of inherent logic. This logic or rationale can be related to structural performance, such as minimal surface development, where gravity becomes the main form-giving agent. The logic can also be any other environmental factor or acoustic performance that informs the geometry of the design. In the context of this thesis, the focus is placed on forms, or more explicitly envelopes, that carry an intrinsic geometric rationale in relation to their manufacturing, construction or performance. These forms are referred to as complex. Complex surfaces can be created through an assembly of standard analytical shapes or through modelling using a constraint-modelling mode, such as translations or revolutions (Glymph et al. 2004). They are default, pre-rationalised shapes.

The term 'freeform' in this context refers to designs that are created without constraints. They are, for instance, physically sculpted and then digitised. An example of these would be the design of the Guggenheim, by Gehry and Partners (Shelden 2002). A freeform, nowadays, is mainly digitally modelled from the beginning using non-uniform, rational B-splines, commonly known as NURBS. Technically speaking, NURBS offer one common mathematical form for both standard analytical shapes and freeform shapes. An example is Wolfgang Tschapeller's design for a hotel building in Vienna (Figure 15). In a freeform, every local part is typically a unique component. The manufacturing and construction of individual components is currently still more expensive. For cost reasons, most freeforms need to be post-rationalised to be realised. A freeform might turn into complex forms when post-rationalised or into a hybrid form.

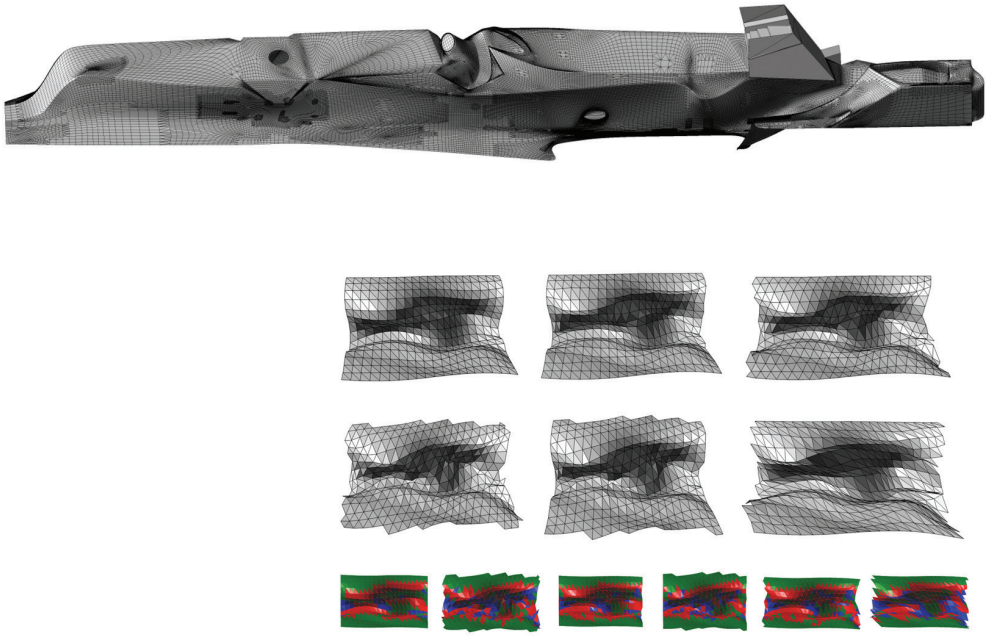


Figure 15
Hotel design, Vienna, by Tschapeller Architects. Design (top) and tessellation studies (bottom).

Hybrids are designs which carry features of different geometric organisations. For example, a design is a hybrid if it is assembled from both freeform elements and complex form elements, such as the Museum of Transport in Glasgow (page 90). Another example is a complex geometry which is subdivided with an order of fragmentation which does not comply with the order of the global form, even though it might be rational as well. A hybrid can also be a freeform which is tweaked to carry a somewhat rational subdivision, such as the façade for the OVO Hilton project in Wroclaw, which is described in the Case Studies (page 83).

3.4. Post-rationalisation versus pre-rationalisation

In the engineering of non-orthogonal form designs, a distinction is made between post-rationalisation and pre-rationalisation. Dr Roland Hudson (Hudson 2010) describes the two different approaches of rationalisation in the following way: "Post-rationalisation is a top down-approach where the final geometry is defined and the parametric design task is to find rational geometry that gives a very close match. Pre-rationalisation is a bottom-up or generative method where the parts are defined and building geometry is a result of combining these". An example of a post-rationalised design is Foster + Partners' Sage Music Centre in Gateshead (Figure 16). The building hull was initially designed as a freeform surface, but due to budgetary constraints it was post-rationalised as a series of tangent toroidal patches. This allowed the surface to be clad with planar quadrilateral stainless steel and glass elements.

Figure 16
Sage Music Centre.

An example of a pre-rationalised design is Foster + Partners' headquarters building for Swiss Re, '30 St Mary Axe' (more commonly known as 'The Gherkin'), which was engineered by Arup. Apart from the study into the relationship between the local diamond shape and the catalogue of the corresponding global forms, which is described in the documented conversation with Prof. Keith Ball (page 38), there was also a mechanical innovation made to control the cost of the cladding. The diamond shape and the necessary framing only repeat at the same floor level but differ between the floors, so the architect invented a universal hinge framing, which allowed the rotation of the angle of the metal mullion section according to the angle of the diamond shape. This way the need for the manufacture of individual connections was diminished. Other examples of pre-rationalised designs are the Al Haramain High Speed Rail stations and Zagreb Airport, both designed by Foster + Partners and engineered by Buro Happold (Figure 17).

Figure 17
Pre-rationalised buildings. Swiss Re Headquarters, one of the Al Haramain High Speed Rail stations and Zagreb Airport.

3.5. Tessellation – reverse engineering intricate form

Every form can be built, including a freeform, without rationalisation. The discretisation of a freeform into building parts, which are coherent with the nature of a freeform, is likely to produce elements which are all unique; an irrational form produces irrational parts. The drawback of not rationalising a freeform is that the production of one-off and compound custom parts, such as doubly curved elements, is still expensive. However, when a freeform is not rationalised there are aesthetic benefits, as a homogeneous freeform design is produced on a local and a global level.

The challenge in the tessellation of complex forms and freeforms arises when applying a geometric rationale to the fragmentation of those forms retrospectively. This is clearly the case when placing a regular grid or mesh pattern onto a building hull that is different from its geometric built-up.

The problem of tessellation is often reduced to wanting to achieve a quadrilateral subdivision and/or planarity, for reasons explained later in this section. In reality though, the task is a multi-dimensional one; the measures of rationalisation to one of these dimensions might have an unwanted effect on another. It is desirable to establish and understand the relationship between the two-dimensional outline, thus the shape of the individual element, its articulation in a three-dimensional space, its material offset and/or its support structure, and the interface to the neighbouring parts.

3.5.1. Approximation

When the fragmentation of a form in a post-rationalisation process is not in accordance with the intrinsic logic of the geometry, it determines that the design surface can only ever be approximated. To holistically apply the rationale, either the

original geometry has to be modified or it serves as a reference for a new model. The rationalisation process ideally describes a given design surface within the limits necessary to maintain the design intent.

3.5.2. Triangulation

Planarity is essential if the final building hull material will be cut from sheet material, and forming the sheet material into single or doubly curved configurations proves too expensive. A straightforward approach to the regular tessellation of an intricate form using planar elements is through triangulation, as by definition every triangle can be planar. When using triangulation, it is often the case that no changes need to be made to the design surface. Every surface can be approximated using triangles, whether it is a complex or a freeform surface. However, depending on the set of objectives for the design, triangulation might not be the right choice. For example, triangulation in glass-steel structures requires more mullions than four-sided tessellation, thereby increasing the amount of steel used, which is likely to make the structures less transparent. The nodes can also become very complicated, depending on the surface geometry that has been triangulated.

3.5.3. Planar quadrilaterals

A popular alternative to triangles are quadrilateral meshes with planar faces. Rectangles (as opposed to triangles) are interesting, as they allow for higher transparency when clear materials are used. Less cutting wastage is produced and the sanding of the edges requires fewer machining operations. Furthermore, fewer mullions, connections and nodes are needed, which affect the cost.

The rectangle, however, is not planar by default. One way to produce a surface that can be subdivided using planar rectangles is through translation. Translation surfaces can be doubly curved and yet be defined by a quadrangular planar mesh. A translation is a surface that can be generated from two section curves, translating one along the other; each then in turn becomes the generating curve (generatrix) and the direction-defining curve (directrix). The curves are swept along each other producing spatial, parallel curves. This process creates a mesh which can be filled with planar four-sided surfaces (Figure 18). A wide range of forms can be created as translation surfaces, such as the Hippo House at Berlin Zoo by the architect Jörg Gribl and the engineers Schlaich Bergermann & Partners (Figure 19). Schlaich Bergermann & Partners collaborated with the architects Gehry Partners on numerous projects using translation surfaces to approximate initial freeform designs (Glymph et al. 2002).

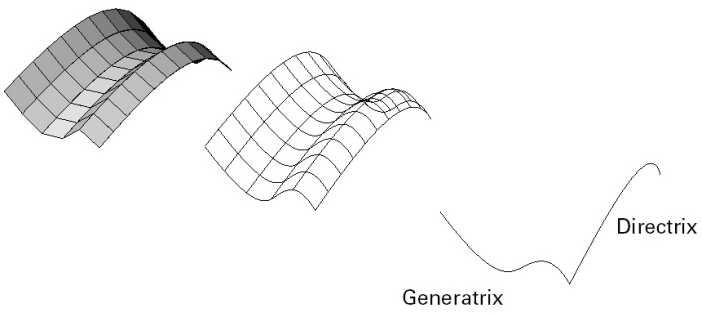


Figure 18
Geometric principle of translation surface.

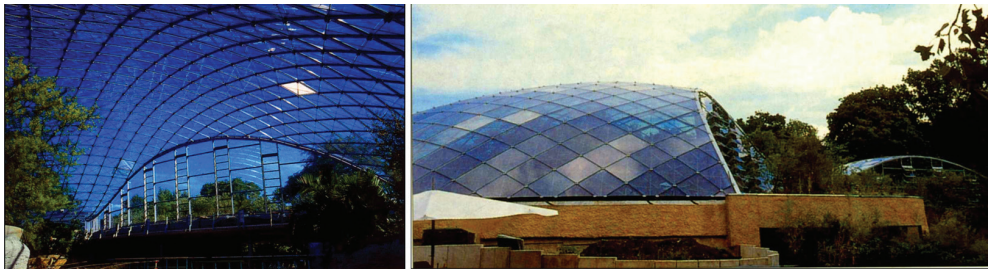


Figure 19
Hippo House. Glass dome of the Hippo House at the Berlin Zoo as a translation surface with a planar quadrangular mesh.

3.5.4. Planar quadrilateral conical meshes

Liu Yang et al. (Liu et al. 2006) developed an algorithm that looks at two objectives: planarity and surface offset. This is interesting because the necessary offset of the material and/or structural support system is very important, but is often not considered. The process optimises a quadrilateral mesh to become a conical mesh with planar quadrilateral panels, which can be orthogonally offset to define the material thickness and the support structure (Figure 20). The work addresses two problems: planar quadrilateral meshes and physical offset production from elaborate envelope designs. The problem in producing the physical offset is described in detail in the High Court of Justice and Transport Museum projects in the case studies of this thesis.

The drawback of this method is the conical shape of the offset cell, which makes it necessary for the edges to be cut and sanded at an angle. The process is a very valuable tool that allows the detailing of intricate glass structures. So, akin to triangulating freeform or planar quadrilaterals in translation surfaces, the complexity is taken out of the form and its three-dimensional definition (as it is planar), and transferred into its shape outline and connection to neighbouring parts.

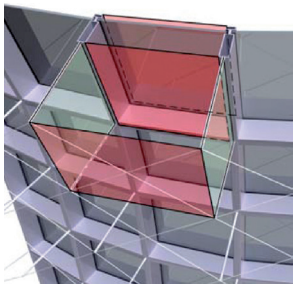


Figure 20
Conical mesh. Illustration showing the conical mesh, which accommodates planar glass elements and connections (Liu et al., 2006).

3.5.5. Same grid member length

A less investigated option for economic tessellation is repetition. One example of repetition is identical element size. This reduces manufacturing steps and potentially eases construction. SMART Form (SMART) is software developed by SMART Solutions, a specialist in-house team at Buro Happold developing custom optimisation and simulation routines. SMART Form can produce a quadrilateral grid having members of equal length by employing an iterative process. The process reliably produces the same-length members, but necessitates a wide range of different intersection angles. At the time of writing, the routine produces an uneven boundary due to the shuffling of the members.

3.5.6. Approximation using a small number of rational shapes

A different approach is to maintain the criteria of repetition, but to allow for single and doubly curved elements. As opposed to planar elements, the advantage of curved panels is that they create inter-panel continuity when used for approximating freeforms. Eigensatz et al. (Eigensatz et al. 2010) propose a multi-criteria optimisation algorithm to approximate freeforms, allowing only a small number of rational panel shapes, while controlling the deviation from the initial design. Eigensatz et al. address the efficiency of reducing the number of necessary moulds and production steps to manufacture the panels. The input is any freeform surface with an arbitrary curve network. The curve network serves as a reference for the intersection curves of the adjacent panels. This creates freeform surface patches, which are approximated by five panel types: planes, cylinders, paraboloids, torus patches and general cubic patches. The deviation from the reference surface, and the position and the kink angle between the adjacent panels are managed simultaneously.

3.5.7. A pitfall in rationalisation

Geometry is called pre-rationalised, in relation to tessellation, if the constraints which govern its development consider specific material properties and manufacturing techniques, and if these considerations are then reflected in the actual fragmentation. One danger is to create a pre-rationalised shape but then apply a different rationale to its tessellation pattern. Pre-rationalised complex designs are only sensibly coherent in their rationale if their fragmentation follows their geometric built-up. The example of the Elephant House in Copenhagen (Figure 21), which was designed by Foster + Partners and engineered by Buro Happold, explains this. The roof geometries were two toroidal patches. The patches were cut at an angle that demanded the surface subdivision to run in a peculiar pattern, and were disadvantageous with respect to structural performance and manufacturing techniques. The architects desired both a regular appearance of the subdivision and the planarity for the glass panels. Therefore, the patches were finally cut out to be perpendicular to the torus' isocurves, and the tessellation followed this organisation.

This example suggests that with complex forms, where the rationale relates to manufacturing, inherent geometric logic should be used as a guideline for fragmentation in the first place.

Figure 21
Elephant House, Copenhagen Zoo, designed by Foster + Partners.

3.5.8. Computer-aided design and manufacturing (CAD-CAM)

Developments in computer-aided manufacturing (CAM) respond to freeform designs in architecture by custom manufacturing directly from the design model. However, they have not solved the problem of cost in freeform manufacturing. CAM processes were primarily developed for industries where an initially expensive manufacturing set-up is offset by the large quantities in which these products or parts are produced. For one-off architectural parts, it continues to be costly.

CAM techniques which build up material (printing) or chip it away (cutting and milling) are mainly used for producing tools or moulds for the repeated processing, casting and forming of the product or part. Three-dimensional printing was developed for prototyping, i.e. testing sequences in product development. The materials in printing processes do not yet conform to the requirements of the building materials.

According to the experience of the author, at this point in time, only two-dimensional CAM processes are inexpensive. A quite radical reduction of a form to a lamella structure of planar ribs allows for the simple manufacturing of parts; the ribs are nested on a sheet material, and cut by a laser cutter or water jet. The complication, however, is transferred from the shape of the structural elements to the interface between the structure and the cladding. Therefore, it is often applied to architectural design models or to a number of pavilions which exhibit the pure structure without a closed skin, such as the [C]space Pavilion 2008, designed by Alan Dempsey and Alvin Huang. The Pavilion was the winning entry in the 10-year anniversary competition of the Design Research Laboratory (DRL) at the Architectural Association in 2007. It was put up in Bedford Square and opened to the public from March to July 2008. The engineers were Adams Kara Taylor (Figure 22).

Figure 22
[C]space Pavilion.

3.5.9. Summary of tessellation problem

The following list catalogues the considerations in the fragmentation task:

- Smoothness is achieved when the size of the describing element varies according to the curvature of the design surface. Where the curvature is steep, the elements describing this curvature need to be small.
- All shapes can be described by triangles.
- Triangles are structurally self-efficient shapes; they are less affected by torsion than four-sided elements.
- Four-sided elements are often preferred to triangles, because they usually produce less cutting wastage, fewer processing steps, higher transparency and fewer mullions. Four-sided elements, however, create a challenge when planarity is desired. Not every surface articulation can be described with planar four-sided elements.
- Every surface which is created as a translation surface can be tessellated with planar four-sided elements, when the construction curves are used for defining the fragmentation.
- Repeating elements is interesting, as it reduces cost by saving form work and manufacturing steps, and usually eases construction.
- Close attention must be paid when complexity is taken out of the articulation of the elements, as it is likely to be transferred to the relationship between neighbouring parts, thus to the connections and/or nodes.
- The use of tolerances in the framing of panel elements to balance misfits in the detailing of the tessellation is possible, but not advisable, as it produces odd reflections and distorted visual proportions.
- A design surface not only needs to be subdivided, but it also needs to be offset, which defines the material thickness and/or the support structure, as well as the interface to the surrounding elements.
- The tessellation task is a multi-dimensional problem; the outline of a local geometry and its form and interface to the neighbouring parts are associated features. If one is changed, the others are likely to be affected too.

3.6. Deductive versus inductive design – designing with constraint geometry

In the current paradigm, designs are generally reverse engineered. Surfaces are retrospectively broken down into buildable parts. The opposite of this realm is the generation, growth or assembly of forms with discrete geometries, or the use of a geometrically constrained modelling mode.

3.6.1. Modular designs

Modular designs are discrete. In architecture they mostly refer to the assemblies of components which are the same or highly similar in appearance, with a standardised connection interface. Examples of modular designs are the Al Haramain High Speed Rail stations (Figure 17), the Queen Alia Airport in Amman and Stansted Airport (Figure 23), all designed by Foster + Partners and consisting of self-sufficient structural support roof units. Another example of the assembly of form is the visually complex interior wall design, P_Wall, by Matsys in 2009 (Figure 23). The wall is built using a tile system; the articulations of the tile surfaces are unique, while the frame and connection conditions are fixed. The tiles are produced using fabric as formwork that is manipulated during the curing process of the material (plaster). As changes to the formwork do not affect the geometric definition of the boundary, the components can be assembled easily.

Zagreb Airport, described in Case Studies (page 102), is a space frame system where structural components are clustered to form the roof. Its visual complexity is achieved by layering different, simple 2D grid patterns, and varying opaque and transparent cladding panels. The named modular or componential bottom-up designs are all relatively two-dimensional in terms of their supporting framework, but they exhibit variety and complexity despite their geometric constraints. They can easily be extended and offer unproblematic maintenance.

Figure 23
Modular designs. Queen Alia Airport (left), Stansted Airport (middle) and (right) P_Wall (Andrew Kudless, 2009).

3.6.2. Designing with constraint geometry sets

Two examples of controlled geometry in design can be seen in works by Antoni Gaudi, a Spanish Catalan architect (1852–1926), and Eladio Dieste, an Uruguayan engineer and architect (1917–2000).

Prof. Mark Burry (Burry 2002) makes a distinction between Antoni Gaudi's freeform approach and the so-called 'Rational Period' of Gaudi's later work. Gaudi's designs from both periods have an expressive appearance, but differ fundamentally in how they were conceived and managed. For example, the early freeforms were sculpted in situ, as no common means to describe, plan and communicate freeform existed at the

time. This meant they needed constant on-site supervision, which was not the case in the 'Rational Period'.

The Rational Period was given this name because it follows the concept of fairly simple mathematical geometries. The entire design is constructed from ruled surfaces; through different curvature degrees and elaborate intersections, the design gains its visual complexity. Burry states that it is a formal complexity which is manageable, as all form elements can be described by straight lines and through a second-order mathematical formula. In this way Gaudi ensured that his work could be documented, and thus communicated and constructed with common building techniques of the time. His supervision, therefore, was not required during the construction.

Mark Burry, as well as many other researchers, took on the task of analysing Gaudi's Sagrada Familia in Barcelona, in order to predict what the unfinished part of the building should look like. Burry has published an elaborate account of the work, in which he states that only the rationale behind the formal complexity of the Sagrada Familia allows him to continue the work on it, which can be read as Gaudi's realisation that this project was likely to exceed his lifetime.

Figure 24
Design prediction within a constrained geometry framework. Mark Burry's studio is working on the completion of Gaudi's Sagrada Familia (Prof. Mark Burry, 2006).

Working in Uruguay and Brazil, Eladio Dieste gained recognition for his elegant masonry shells. He built them from locally manufactured brick, using only a small number of different brick shapes (Figure 25). The forms of the global shells, mostly Gaussian vaults, are what resist structural forces. Dieste created a technique to pre-stress masonry work, which allowed him to construct shells with a single layer of bricks. He also invented a particular movable formwork to support the construction (Pedreschi 2000). Limiting his work to a single material product with a small number of local shape variables, Dieste established extensive knowledge of the handling of the material and explored the catalogue of possible structures, which were derived from the association between the local component and the performance-driven global geometries. One could say that he established an associative material system consisting of the local bricks, their material and shape, and the assembly of these to form larger structural shells.

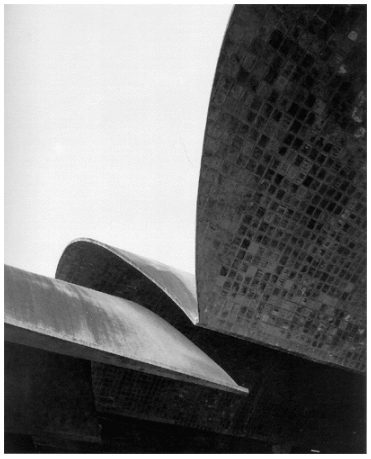


Figure 25
Massaro Agroindustries. Pre-stressed brick shell by Eladio Dieste.

3.7. Summary

At this point in time, most non-orthogonal envelope designs in the building industry are reverse engineered and, in a majority of the cases, the order of their fragmentation is different from the logic of their geometric built-up.

This chapter introduces objectives which guide the process of reverse engineering elaborate surfaces, and brief descriptions of various approaches that can be used for tackling the task of discretisation are provided.

Non-orthogonal forms are divided into three main types: freeforms, complex forms and hybrids. This differentiation helps to analyse the relationship between the global forms and their potential subdivisions.

The definition of the local parts which constitute a global geometry is recognised as a multidimensional problem. The two-dimensional outline, the shape of the local element, its articulation in a three-dimensional space, its physical offset (material thickness and/or supporting structure) and the interface to neighbouring parts (edge characteristics, connections and nodes) are the associated features. Changing one is likely to affect the other and these local features are, in turn, interlinked with the global geometry. Not every local rationale can be propped onto an irrational global geometry without having to change the global form.

It is commonly assumed that approximating an irregular shape with a regular tessellation results in a cost saving, but this is not necessarily the case. Overlaying different orders of geometry usually produces a shift of the problem and not an actual elimination of it.

Manufacturing and construction objectives may or may not play a role in conceiving a design. When processing a design, forms may be fragmented in line with the order of their intrinsic geometric logic or as hybrids, where a different order is overlaid. The decisions would preferably be made based on all the encompassing information. The question may be asked that if geometry is rationalised in retrospect, why then is it not created as a rational form in the first place. One answer might be that these considerations are simply not part of the aesthetic design conception.

Everything can be built, and when fragmentation is necessary but not part of the aesthetic interest, numerous post-rationalisation techniques are available to solve

problems of detailing. Early approaches for tackling elaborate designs can be avoided, as in the case of the Experience Music Project in Seattle by Gehry Architects. The intricate building form is an offset from a regular orthogonal structure and the cladding panels are partially bent into place on site, leaving a patchy appearance.

However, these types of projects, and in particular Frank Gehry's designs, triggered advancement in reverse engineering. They form the catalyst for a knowledge base about intricate forms, their design and engineering.

This chapter also presents an alternative realm, where a design is created using a constraint geometric mode of modelling or it is inductively generated from discrete local geometries (material products or grander modular entities). These concepts aim for design integrity, and a coherence between the local and the global order of geometry. Pre-rationalisation in relation to manufacturing and construction seems to demand an element of innovation. The next chapter, Case Studies, documents examples of real projects in which the objectives and challenges described here come into play.

CHAPTER | 4

CASE STUDIES

4

CHAPTER | 4

CASE STUDIES

4.1. Introduction to case studies at Buro Happold

An essential part of this study is the involvement in active projects of the sponsoring company Buro Happold (Figure 26). This involvement in projects allows for the understanding and recognition of recurring themes and problem patterns in the design and engineering of geometrically intricate buildings. The involvement initiated the identification of the research domain and aided the creation of control parameters for the proposed design system.

This chapter documents a selection of projects which the author has been working on and which are immediately related to the research. A schematic table is presented listing six projects and their names, the tasks and subtasks. The overview is followed by a detailed account of each project. The chapter closes with the identification of objectives which are taken forward to the research project.

At the beginning of the project work of this study, which coincided with the setting up of the GG group in Buro Happold, every project brief was a pilot study and each solution was developed from scratch. Over time, problems to be solved reoccurred, and this way, solution methods could be reused on different projects. The type of models produced varied from fixed to partially or fully parameterised models. Partially parametric models were either created using GenerativeComponents, a parametric modelling plug-in for MicroStation, which at the time was still in its Beta version, or scripting with program-embedded languages for AutoCAD and Rhinoceros. Fully parameterised models were produced as C# plug-ins to Rhinoceros.

Project	Year	Architect	Location	Acronym
OVO Hilton	2009	Broadway Malayan Architects	Wroclaw, Poland	OVO
Museum of Transport	2005/2008	Zaha Hadid Architects	Glasgow, UK	MT
High Court of Justice and Supreme Court	2007 – 2008	Foster + Partners	Madrid, Spain	HCJ
Zagreb Airport	2008	Foster + Partners	Zagreb, Croatia	ZA
New Holland Island	2006 – 2007	Foster + Partners	St Petersburg, Russia	NHI
National Academy of Arts	2006 – 2007	Snøhetta	Bergen, Norway	NAA

Project	Task	Material [1]	Objective	Procedure	Problem	Project Deliverables	Involvement of the Author
OVO	• Façade tessellation	• Glass, polymer and aluminium	• Preservation of design • Cost rationalisation • Reflecting wall and floor partitioning in the façade tessellation • Planar façade panels	• Geometry analysis • Development of inductive and deductive rationalisation options: (inductive) remodelling the building hull using translation surfaces, (deductive) developing an iterative parametric procedure to adapt the form of the hull and tessellation of the façade using building tolerances • Generating alternative façade models	• Creating structural and internal surface offset from freeform • Forcing planar façade elements onto a freeform building hull with the constraint of floor and wall lines	• Organisation of 2D sections, plans and 3D building hull model • Geometry analysis report and proposal for rationalisation • Models representing building hull parts to demonstrate rationalisation options • A range of models, each with a different tolerance value and consequently different form and façade grid	• Client meetings • Organisation of project data • Geometry analysis and report writing • Developing the proposal for rationalisation • Creating models, demonstrating the proposed alternative rational approaches
MT	• Envelope structure • Internal envelope skin • Façade tessellation	• Façade: Aluminium or steel • Structure: Steel	• Preservation of design • Cost rationalisation • Form coherence between all layers • Flat panels of equal size for façade tessellation	• Geometry analysis and rationalisation • Structural model • Construction model • Consultation on internal surface definition • Façade tessellation	• Creating structural and internal surface offset from freeform • Bringing a regular façade tessellation onto an irregular form	• Geometry analysis and report • Proposal for rationalisation • Consultation on the making of the design model • Principal structural model • Principal construction model • Parametric model for the fish-scale façade design	• Client meetings • Geometry analysis and report writing • Development of proposal for rationalisation • Supporting the architect to make the design model • Creating a principal structural model • Creating a parametric model for the fish-scale façade design
HCJ	• Roof structure	• Glass and steel	• Preservation of design • Offset which corresponds to the design form	• Development of an automated process to develop the structural offset from the design surface, including a detailed definition of nodes and rods • Production of principal structural model	• Creating structural offset from freeform. • Defining connections within roof build-up tolerances	• Principal model • Principal construction model • Project report	• Writing the report on the design development of the roof
ZA	• Consultation for modular system design • Structural detailing	• Steel and glass or polymer composite	• Visual complexity, despite high repetition of roof modules • Extendibility	• Researching different grid options and how they could be overlaid • Design of the structural component and construction sequence	• No problem, interesting study into modular grid-related roof system and bottom-up, as opposed to reverse-engineered, procedure	• Concept sketches • Competition report on structural build-up and construction sequence	• Concept development
NHI	• Roof structure	• Glass and steel	• Cost rationalisation • Optimisation of material usage • Preservation of diagrid • Fixed height of roof • Maximum transparency	• Relaxation of given roof model • Form-finding of roof form from scratch, within defined support boundaries • Grid layout optimisation: comparative study of different directional diagrid bracing • Study on structural grid layout freed from the diagrid using an in-house research code	• The diagrid and subsequently necessary bracing conflicted with the objective to use a minimum number of members and minimum member thickness. In particular, the architect initially insisted on the less structurally efficient direction of the bracing	• Report on roof form optimisation • Report on grid layout study • Provision of alternative form and grid layout models (for light studies, that the architect conducted) • Principal model	• Client meeting • Structural analysis of initial model • Optimisation of the initial model • Form-finding new model from scratch, which becomes the principal structural model • Comparative analysis of developed models • Report writing on the design development of the roof
NAA	• Roof design • Roof structure • Roof tessellation	• Glass and steel	• Interesting roof form • High transparency • Minimum cost	• Structural form optimisation: Informing the roof articulation by its load bearing to improve structural performance. This way less material is needed and cost is reduced. Digital form relaxation of a given roof shape. • Production of design alternatives: form-finding a roof from scratch and varying boundary conditions • Comparative analysis of design alternatives • Structural grid definition • Optimisation of grid subdivision: informing the tessellation by the warping factor tolerance to produce four-sided elements, to minimise triangulation in the surface and thereby reduce material and cost.	• No problem here as the architects were curious to explore what the form of the roof wanted to look like, i.e. if it could drop into shape using gravity as a form giving factor within the given boundary constraints; informing the roof form entirely by structural performance allowed for a maximum material saving	• Progress report on optimisation • Design alternatives • Principal structural model	• Optimisation of given roof form • Form-finding the roof from scratch, which becomes the principal structural model • Comparative analysis of developed models • Writing the report on the design development of the roof

Figure 26
Project table.

Note: The final design and material used for a project might vary from the above description. Some of the work is done in a specific time window of a long design process.

4.2. OVO Hilton

4.2.1. Project description

The Hilton Wroclaw project comprises a new hotel complex in Wroclaw, Poland. It was commissioned by Wings Property and designed by Gottesman-Szmelcman Architecture and Broadway Malyan Architects (Figure 27). The Generative Geometry group at Buro Happold, within which the author was based during the study, was consulted to help post-rationalise the façade. Both architectural clients wished to produce a planar tessellation of four-sided elements on a freeform building hull. The tessellation lines were to follow the floor levels, respecting wall partitions. Different possibilities were proposed in which the principal design model would be recreated from the bottom-up, with the rationale of the façade objective in mind. All options used the principle of translations to produce a planar tessellation, although in the end a top-down approach was required. The architects' model was post-rationalised employing a custom-made computer code, in order to force planarity using building tolerances. The parametric code allowed the control of the level of deviation from the design model.

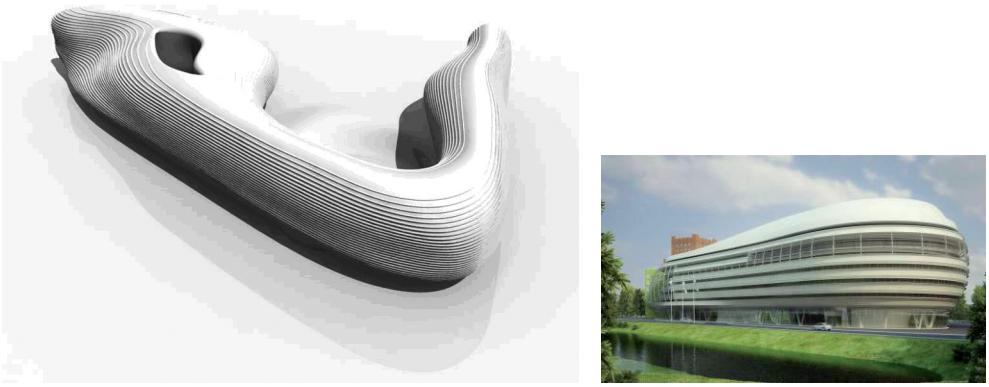


Figure 27
OVO Hilton (Gottesman-Szmelcman Architecture and Broadway Malyan Architects, 2009).

4.2.2. Project brief – planar façade tessellation with directional constraints

The architect created the outer hull of the hotel as a doubly curved, freeform design. The brief for the façade was to achieve planar four-sided elements, to enable conventional glass construction. The gridlines for the subdivision of the façade had to reflect the floor levels and wall partitioning. It was decided that the material for the mullions would be aluminium. A combination of glass and an additional opaque material, such as a metal or a polymer, was considered for the surface elements. Despite trying to resolve the problem with multiple tessellation tools in Rhinoceros and CATIA, the architect only achieved planar tessellation in some areas; most elements remained doubly curved. Figure 28 shows a part of the front façade of the architect's model, which indicates a tessellation produced by a CATIA plug-in. The model is colour-coded, showing the curvature of the individual panels: grey specifies a planar element; green and red are curved elements, whereby green has less curvature and red has more curvature. The model allowed local analysis of the given geometry but did not produce the desired planar tessellation (Figure 28). Buro Happold's GG group was consulted to analyse and optimise the façade, to ideally attain planar elements which would correspond to the floor levels and wall partitions.

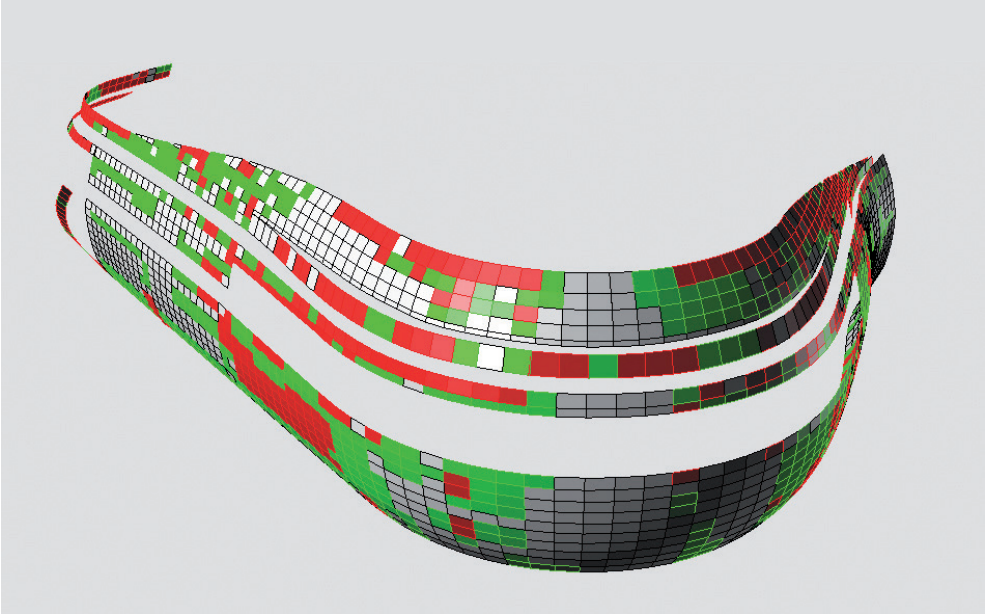


Figure 28
CATIA model. Part of the front façade of the architect's model showing a tessellation produced by a CATIA plug-in.

4.2.3. Process

Design data synchronisation

Before the commencement of the analysis of the design, the reference data had to be verified. This verification showed that the three-dimensional (3D) model and the two-dimensional (2D) floor plans did not match. At one point during the project development, the two independently managed modes (the 3D model and 2D floor plans) were not synchronised; a problem of data management. The architect wanted the internal program and the external building hull to be created in the same model, so that plans and elevations could be extracted from the three-dimensional model. The absence of a principal model demanded the synchronisation of all the data as a starting point. Floor plans were inserted into the three-dimensional model and updated according to the perimeter of the building.

False impression of geometric rationale

When analysing the front façade of the main building, it became evident that despite the appearance of a simple torus shape, with a scaling of the radius in the front part (varying conical shape), the form was in reality a freeform design. All isocurves of the side wings were different in shape.

Hence the form of the building hull was not designed with the constraints of the tessellation in mind. Given that the rationale of how the model was created was not available, this made it harder, but also important, to identify and fix objectives to progress the project. The GG group recommended the rebuilding of the model from scratch, keeping the constraints for the façade in mind while preserving the intended appearance. Three options were proposed. The first two options required the principal model to be rebuilt. The third option only partially remodelled the design from simple translations at the side wings and a toroidal section at the front, to allow for triangles or doubly curved elements in the transition areas and the fan-type end of the right wing.

Possible solutions 1 and 2: rebuilding the model using translation

Both solutions make use of the principle of translations as a means to create complex forms which can be subdivided into planar quadrilaterals.

The first solution starts with the simplest option: to sweep a single section curve vertically along either the upper or the lower floor plan perimeter (horizontal curve) to form a translation surface. The resulting surface could be subdivided into planar quads, referring to both the floor levels and the wall partitions. However, the form would be simplified significantly. The potential loss of the dynamic appearance of the building was the reason that this option was not considered.

The second solution was to assemble the model using different modelling methods that result in planar reticulated structures (Stephan et al. 2003). These methods were dilative translations (simple and scaled translation) for the side wings of the building and one surface of revolution in the middle section (Figure 29). The resulting model could be covered with planar elements. This option was closer to the initial design than the first solution proposed. The disadvantage of this scheme was that the intrinsic mesh of the scaled section would not fulfil the constraint for the façade, i.e. to reflect the wall partitioning and the floor levels at the same time. That is due to the generating curve of a scaled translation, which sizes up or down while sweeping along the direction curve, not producing orthogonal isocurves as shown in the example in Figure 30. Figure 30 a shows the generating curve B1, which increases in size to become curve B2 while sweeping along the directional curve A. Image b presents the resulting surface. Image c shows the isocurves on the surface; the red curves are parallel to each other, while the blue curves follow the scaling of the generating curve B. Any deviation from this constraint would necessitate a change in the overall form, resulting in the need to revisit the internal building program. The architect was satisfied with the current solution for the internal building program and ideally did not want to revisit this. Therefore, this option was only considered for a while and not taken forward.

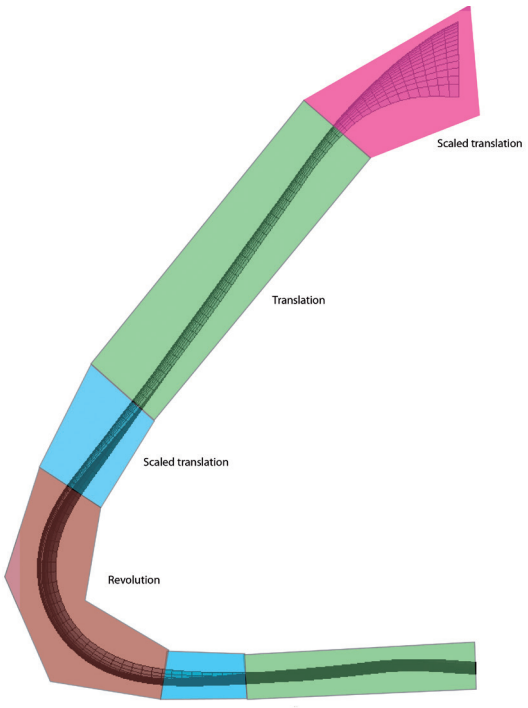


Figure 29
Possible translation types. The coloured fields show a possible division into different types of translation to ensure planar elements.

Figure 30
Illustration of the principle to create a scaled translation surface.

Possible solution 3: allowing for doubly curved or triangulated elements in some areas

A third alternative was to describe the scaled surface areas, either by triangles or doubly curved elements, and only rationalise the side wings to be buildable from planar four-sided panels. The architect, however, did not want to consider a combination of triangulated and rectangular elements.

Solution: working with construction tolerances

The options considered so far did not produce satisfactory results for the architects; the dilemma was that any proposed procedure to produce a tessellation with rectangular planar façade quadrilaterals demanded a change in the building form, which subsequently shifted the floor plans. The solutions considered show the misconception that the given freeform shape of the building could simply be subdivided into planar four-sided façade elements, and reflect wall partitioning and floor levels, without any changes in the global geometry.

The recommended methods at this stage would have reconstructed the model from scratch with the façade criteria in mind. The remaining possibility was to reverse engineer an updated version of the existing model by applying an iterative optimisation process, in which the façade that is represented by a point grid was manipulated within construction tolerances (Figure 31). These tolerances lay at the meeting point of the adjacent panels. The selected façade system had a wide frame that held the individual panels. The width of the frame allowed the panels to be tilted slightly with respect to the bordering panels. This process also altered the overall geometry, but the changes could be controlled by the defined tolerance parameter.

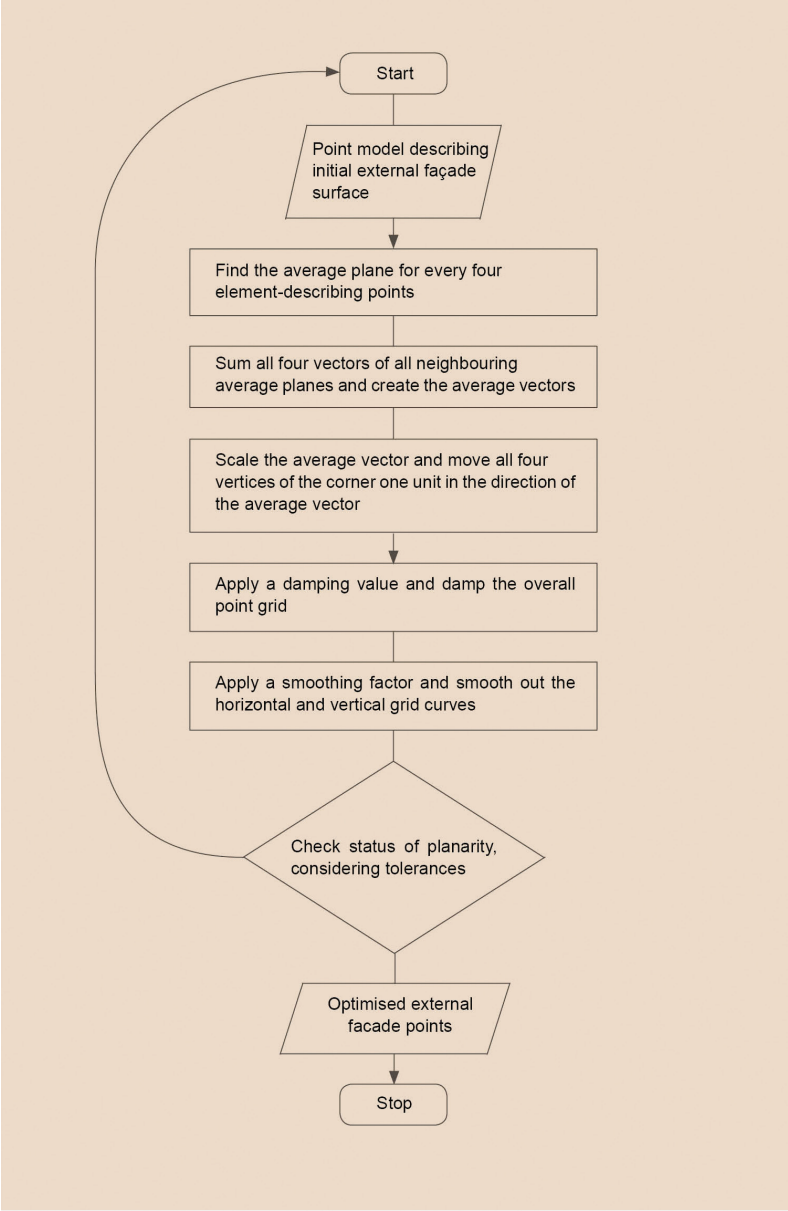


Figure 31
Point grid optimisation flowchart. The process flattens the four-sided panel façade within defined tolerances.

The procedure was as follows: the average plane was found for every set of four points which described an element, all four normal vectors of all the neighbouring average planes were summed and the average normal vectors were created. Then the average normal was scaled and all the four vertices were moved one unit in the direction of the average vector. A damping value was applied and finally a smoothing factor was used, and the horizontal and vertical grid curves were smoothed. The status was checked and the routine was repeated until the result matched the tolerance which was specified up front.

The tolerances per panel and the individual panel size allowed the control of deviation from the initial design model (Figure 32). The automated process allowed the influence of the tolerance parameter to be observed. As the tool could be used on any grid point, the architect could have supplied a completely new design and the same routine could be applied. This program was reused in other projects.

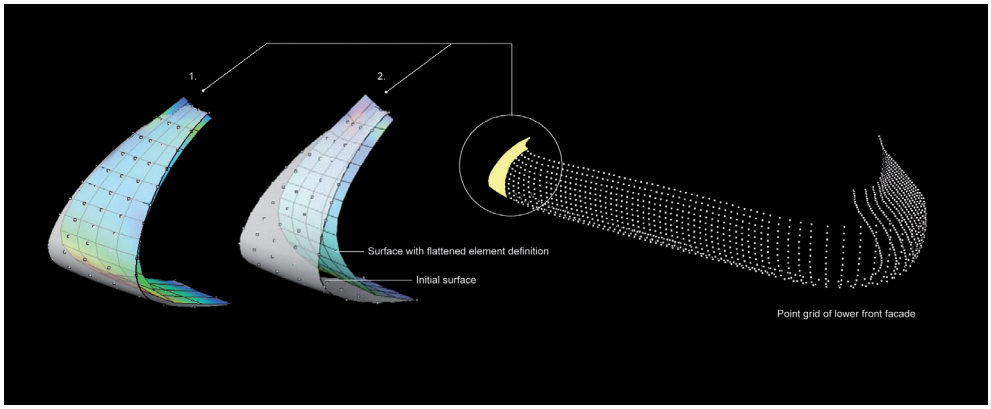


Figure 32
OVO Hilton optimisation process. Diagram of a section of the lower front façade. The numbers 1 and 2 indicate different tolerance settings. With a smaller tolerance, the optimised version deviates further from the original surface.

4.2.4. Summary

The project is an example of a scenario where a global geometric order is meant to be tessellated with a conflicting local order of tiling. It is possible that the client and the architects could have been convinced to recreate the model from scratch, if the group had not proposed a third solution. The project showed that geometric logic is not necessarily the main driver for the client to take design decisions. The architects chose the deductive approach, as they wanted to retain control over the form as a whole. Finally, the design was modified and not recreated.

The automated process allowed for a gradual adaptation of the building form to the desired façade constraints: the matching of the façade subdivision with the floor levels and wall divisions. The smaller the tolerance setting, the greater was the planarity of the panels and the precision of the joining between the panels. The more the form developed towards planarity, the greater was its deviation from the original form. Here, the assumption was that when the tolerance parameter in the optimisation process was at a negligible level, the resulting geometry would produce one of the translation surface options. This means that if an optimum solution is found through manipulation, it must follow a geometric mathematical logic.

In reverse, the bigger the tolerance-setting, the closer the model stayed to the original surface. However, this meant more of the panels did not line up with their neighbouring parts. The process relied on a façade system with wide frames to accommodate the fact that the edges of neighbouring panels did not line up. The wide frame allowed that a planar panel could be fixed in a tilted position, which differed to the position of the neighbouring panel.

Models with different tolerance values were produced and supplied. This enabled the architect to test how much of the internal programme had to change with the changes in the form of the building hull. The final model compromised tolerances of the façade system frames with the necessary construction tolerances. Subsequently, the design did change and the internal program had to accommodate these alterations of the building hull geometry, but the architects were satisfied with the process and the result.

The use of the tolerances of the prospective façade system in the planning stage was not desirable, and the architect had to make sure that the minimum of the construction tolerances was maintained, using a frame façade system with a bigger tolerance built

into the width of its members. The group had two preferences to solve the design problem. One would have been through a more radical, but subsequently consistent geometry, i.e. a tessellation. This could have had either planar elements, through greater changes to the geometry, or looser constraints on the line-up with the internal room partitioning and floor levels, through smaller changes. The alternative idea was to balance changes to the geometry by allowing the panels to be doubly curved.

An interesting observation was that there was a preoccupation with the idea of economic efficiency, by deciding to place a planar tessellation on the freeform building hull in retrospect. The architects had already spent significant resources trying to tessellate the façade before consulting the group. Through the process, the architects learnt that the advantage of planarity in the façade (which meant the elimination of curved elements) produced the disadvantage that all elements would be shaped differently, and therefore all connections would be different; a more cost-efficient option could be achieved through repetition. Then even doubly curved surface elements would not necessarily be a problem, depending on the number of repetitions achieved. It was new to them that the combination of explicit constraints required a particular global geometry and was not compatible with their initial design.

It is interesting to note that the inductive approach to manipulate the form was preferred over a reconstruction of the model, even though both scenarios ultimately caused changes to the floor plans.

Achievements

- Conveying geometric dependencies of local and global geometries.
- Development of an inductive process to iteratively scan and manipulate a grid, to create a planar or quasi-planar four-sided tessellation.
- Production of a model with quasi-planar tessellation within the architect's design constraints, while staying as close as possible to the initial design form.

Relevance for the research project

- Dependency between façade tessellation and the global building form.
- Freeform versus complex form.
- Necessity to communicate pitfalls of post-rationalisation. This means that post-rationalisation is not always possible or that it might compromise a different criterion and shift the cost, not reduce it.
- Planarity versus repetition.

4.3. Museum of Transport

4.3.1. Project description

The Museum of Transport in Glasgow was designed by Zaha Hadid Architects. Buro Happold provided the structural and services engineering. The building shape reflects a number of roof folds extruded at variable angles and bent to form an in-plane zigzag shape. The building height drops from one end to the other. As this drop in height does not lie in one plane, it causes detailing complications at the curved corners of the zigzag envelope fillets (Figure 33).

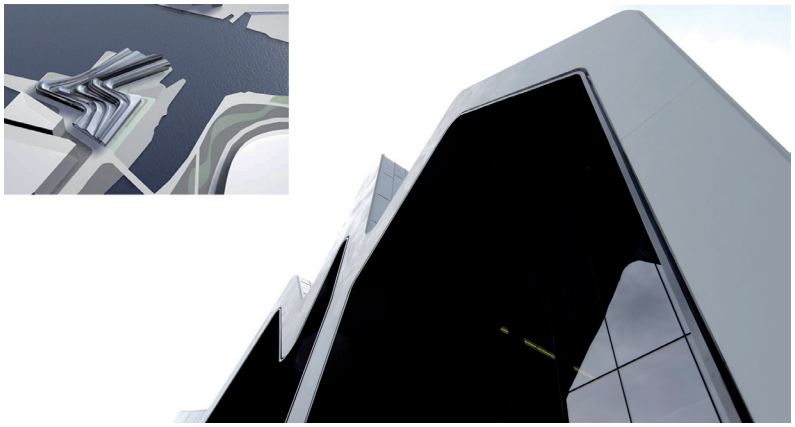


Figure 33
Museum of Transport. Competition rendering (top left) and a section of the build design showing the front facing the water.

The GG group created the structural model from the external design surface, helped to define the inner ceiling surface and developed the initial façade design. The structural model for analysis needed to consist of straight centrelines. The internal ceiling was intended as an offset from the external pleated envelope, in order to express the outer appearance in the inner exhibition space. The initial façade design was intended to be a fish-scale tessellation with metal panels, which was later replaced by a simpler design.

4.3.2. Project brief

The project involvement in the Glasgow Museum design was threefold: first, creating the principal model through the extraction of key geometry, and rationalising and remodelling the initial design; second, analysing the geometry in order to create the structural offsets, as well as the definition of the internal ceiling; third, conducting studies on the fish-scale pattern for the façade.

4.3.3. Rationalisation of design model

The architects provided the external design surface, which was produced in the CAD modeller Maya. Imported into Rhino, the geometry was defined as surface patches showing partial overlaps in places, and gaps between the larger envelope surfaces and the blended surfaces at the folds. Therefore, the geometry needed to be remodelled.

It was difficult to retrospectively understand the process of how the model was created and hence to understand the guiding geometric principles. The design model

was edited by a number of architects, so the elements were manipulated rather than reconstructed. In addition, it was difficult to define the central fold lines on the ridges and valleys of the external envelope, as they were filleted between the adjacent surface areas. As these areas were built up from patches rather than continuous surfaces, extending them to find a continuous intersection curve was not possible. Therefore, unclear geometric parts needed to be re-defined, and the required approval was obtained from the architects.

For example, the definition of the rounded corners, where the envelope changed direction in plan, needed attention. The roof descended from one side of the building to the other, but the individual roof ridges did not lie in a plane. Each individual ridge and valley descended by a constant drop per metre (the descend factor), which meant the round corners did not create tilted arcs but twisted freeform curves instead, as shown in Figure 34. These caused conflicts when describing the fillets, i.e. the smooth transitions between the wall areas and the roof.

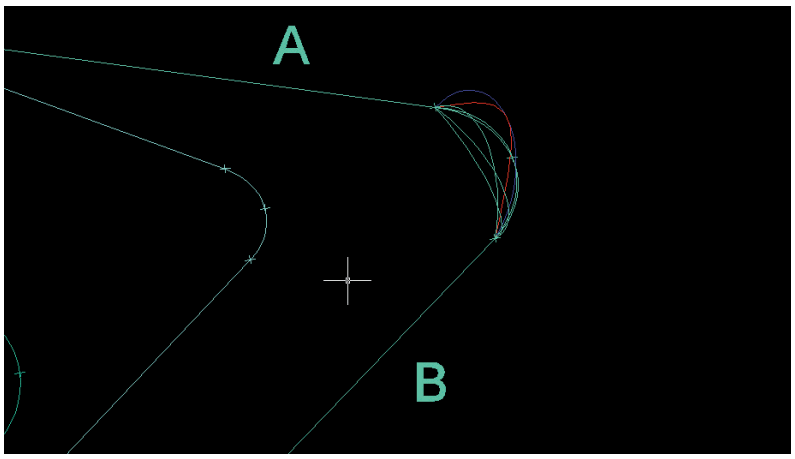


Figure 34
Corner fillets. When filtering the key geometry of the envelope, corner fillets between two straight curves had to be defined.

To ease the ongoing geometric development, the GG group proposed that the zigzag-shaped curves which define the gables and valleys could lie in one or two planes per ridge or valley curve (see A and B in Figure 35). This way, the fillets at the corner would be resolved by the formation of tilted arcs, using three points.

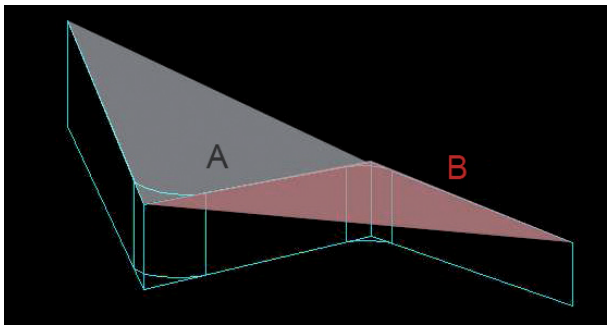


Figure 35
Diagram of the left external wall, proposing that the upper zigzagshaped curve lies in two distinct planes (A and B).

The model was analysed, and the geometric areas eligible for rationalisation were identified. The GG group suggested how certain parts of the model could be created differently. It provided models of the details to be changed, as well as instructions of how to create them. The principal design model was reconstructed by the architects,

and revisited and checked by the group. Some surfaces which appeared straight, but were in fact doubly curved in the initial design, were replaced with straight ones. Complicated doubly curved faces were eventually substituted by translation surfaces, which allowed for a planar subdivision. The proposal to replace the drop of the building height per metre, by placing the ridge and valley heights on sloping planes in order to create tilted arcs where the geometry changed direction, was not taken on. Instead, arcs were fitted into the existing geometry. From the point of view of the GG group, the change to planes would have been geometrically more consistent without disturbing the appearance dynamics. All the proposed changes were discussed, decided and implemented together with the architects.

4.3.4. Definition of structural offset and internal ceiling

The structural model was derived from the design model, and demanded a separate iterative process of the changes and the analysis. The structural model had to be constructed from straight lines for two reasons: firstly, because the final structure was to be made of straight beams; secondly, because the finite element software used could only read polygons. These straight lines represented the centrelines on which to place the structural members. This model also served to extract the information for construction.

It was intended as a direct parallel offset of the outer surface definition given by the architects. The structural model indicated the geometry-defining curves, ribs and bracing.

Once the external design surface was defined, the structural offset and the internal ceiling offset showed complications. A geometric analysis was necessary in order to understand these. A regular offset, normal to the original face, resulted in a geometrically peculiar, and therefore undesirable definition of the structure and the internal ceiling. The reason for this was the drastic changes in the ridge angles (Figure 36 and Figure 37).

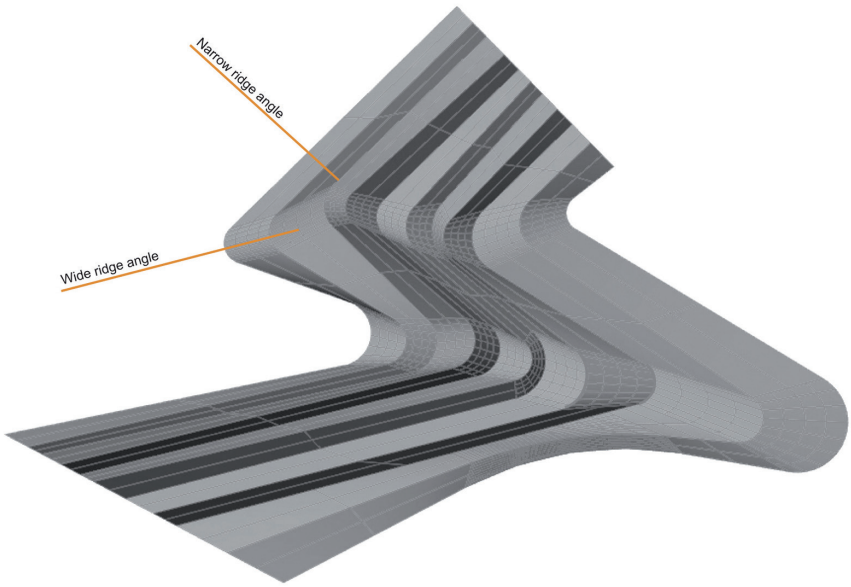


Figure 36
Plan view of MT model. Two areas in the model of the external envelope are pointed out, showing two ridge angles lying close to each other but differing greatly in size

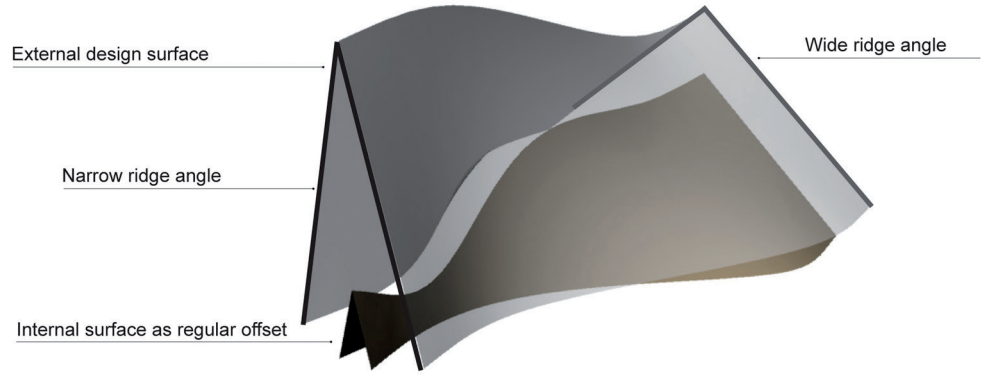


Figure 37
Varying ridge angles. Diagram of a regular, normal offset in a scenario where the ridge angle size changes drastically over a small distance.

Two possible solutions were considered. The first option was to remodel the design, starting from a central surface which would represent the structural layer, and then offset the internal and external envelopes. This reduced the control of the external envelope definition and, for this reason, it was ruled out by the architects, as they wanted to keep the external form unaltered. The second option was to model the internal envelope independently from the external building hull, which allowed the appearance of both to be fully controlled. The latter option was taken forward, and the structure and internal ceiling were modelled separately (Figure 35). In some areas of the structure a regular offset could be created; in other parts the offset had to be modelled separately, to ensure its fit between the internal and the external surfaces. In order to free-span the folded roof, and thereby create a column-free exhibition space, the folding of the envelope had to serve as an in-plane stiffening feature. It was necessary to make a multitude of changes to the structural model, which needed to maintain its position between the two design surfaces, while spanning the width of the building without additional supports (Figure 39).

The cladding was supported by the brackets fixed to the roof built-up of the metal decking on the steel structure; all the brackets were different, and any mistake in them would be carried through and become evident when placing the external cladding.

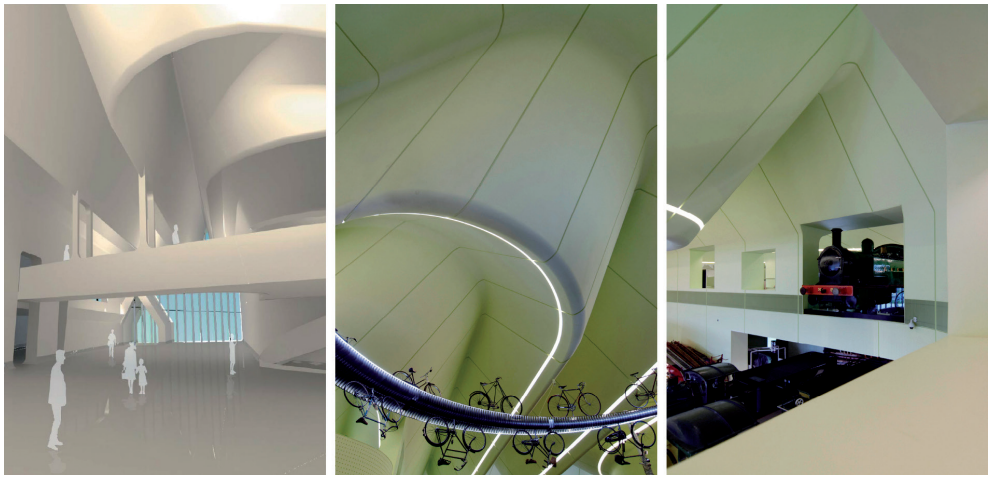


Figure 38
Internal skin. Rendering (left) and final exhibition space (middle and right). The internal envelope appears as a regular offset of the external building surface (Zaha Hadid Architects, 2011).

Figure 39
Exposed structure in the exhibition hall.

4.3.5. Façade tessellation

The initial challenge of the façade system was tiling the complex architectural geometry with a regular pattern. With the aim of attaining a homogenous tessellation of the outer hull, the architects intended to lay out similarly sized rectangular steel or aluminium panels in a fish-scale pattern, thus overlapping. The panels were envisaged to have a pearl gloss finish, which dictated that they could not be processed and therefore had to be used as flat sheet material.

An organic solution for the façade would have been to scale the size of the tiling with the size and curvature of the building, but this was not desired for two reasons. First, it was assumed that it was more cost-efficient to employ the same size of panels, and second, the architects felt a change in the panel size would not be aesthetically pleasing.

The constraints of the design, the repetition of the panel size and the panel planarity were challenging, given the varying widths of the surfaces resulting from the unparallel running of the folds in the building skin. The task was to create a parametric model, which allowed the adaptation of a regular tessellation pattern to the changing underlying surface geometries. The GG group was called in because the architects were struggling to create a visualisation of the façade design, by trying to map the fish-scale pattern onto the global building form. The group suggested solving the tessellation so that it was not only serving the purpose of visualisation, but so that it actually functioned. Producing a convincing visualisation of the tessellation was equally as hard as detailing the facade itself.

The solution explored was to identify a parametric reference grid on which the shingles could be laid out. The question was how to generate a grid on the hybrid geometry whilst addressing two specific constraints on the shingles: they needed to have sufficient overlap with the layer below (to work effectively in providing a waterproof surface) and their rotation was restricted by the position of the adjacent shingle (to avoid the need to cut its corners).

As the overall geometry was constantly changing, a non-parametric model of the cladding would have been inefficient and, above all, time-consuming. Using GenerativeComponents (GC), Bentley's plug-in for MicroStation Triforma (pre-Beta

version), the author created a parametric model allowing the fish-scale pattern to adapt to the formation of the adjacent curves.

Two components were defined in GenerativeComponents: first, a component that generated a parametric grid of points between the two curves and second, a discrete shingle component that would be proliferated on the reference grid. The point pattern of the grid had to function as a reference for placing the shingles in the desired fish-scale layout of the façade design. The grid represented a sidewall surface or a roof area. To place the shingle on the grid, one reference point (grid point) was needed with a local coordinate system, to ensure the correct orientation of the panel (Figure 40). The shingle would automatically orient itself in a specific double-tilt on the grid. Unlike numerous other GC projects, the component did not freely adapt its size and articulation according to the underlying grid. That is due to the fact that the panel was placed by a single point. Also, the grid was constrained and ensured that the shingle size was always maintained. What this meant though, was that the upper border was irregular and the shingles had to be bent over the roof ridge in an undesirable way, both aesthetically and in terms of the . However, these transition areas were not included in the scope of the extended work on the façade.

During the development of the facade, before the setting up of the described GC model, Zaha Hadid Architects decided that the fish-scale pattern was too expensive, and Buro Happold Façade stopped its further development.

The ability to place a regular reference grid onto an irregular building form is a recurring problem in the tessellation of intricate form and thereby served as a good exercise. The author decided to continue with the development, outside of the project commission.

The actual built façade became a continuous skin of assembled, perforated zinc sheets, which replaced the fish-scale façade design. Each of the perforated panels had to be handcrafted, so the cost was still high. However, the architects wanted to ensure that the finished envelope surface was smooth. In this way the decision for a generally less complicated layout was taken.

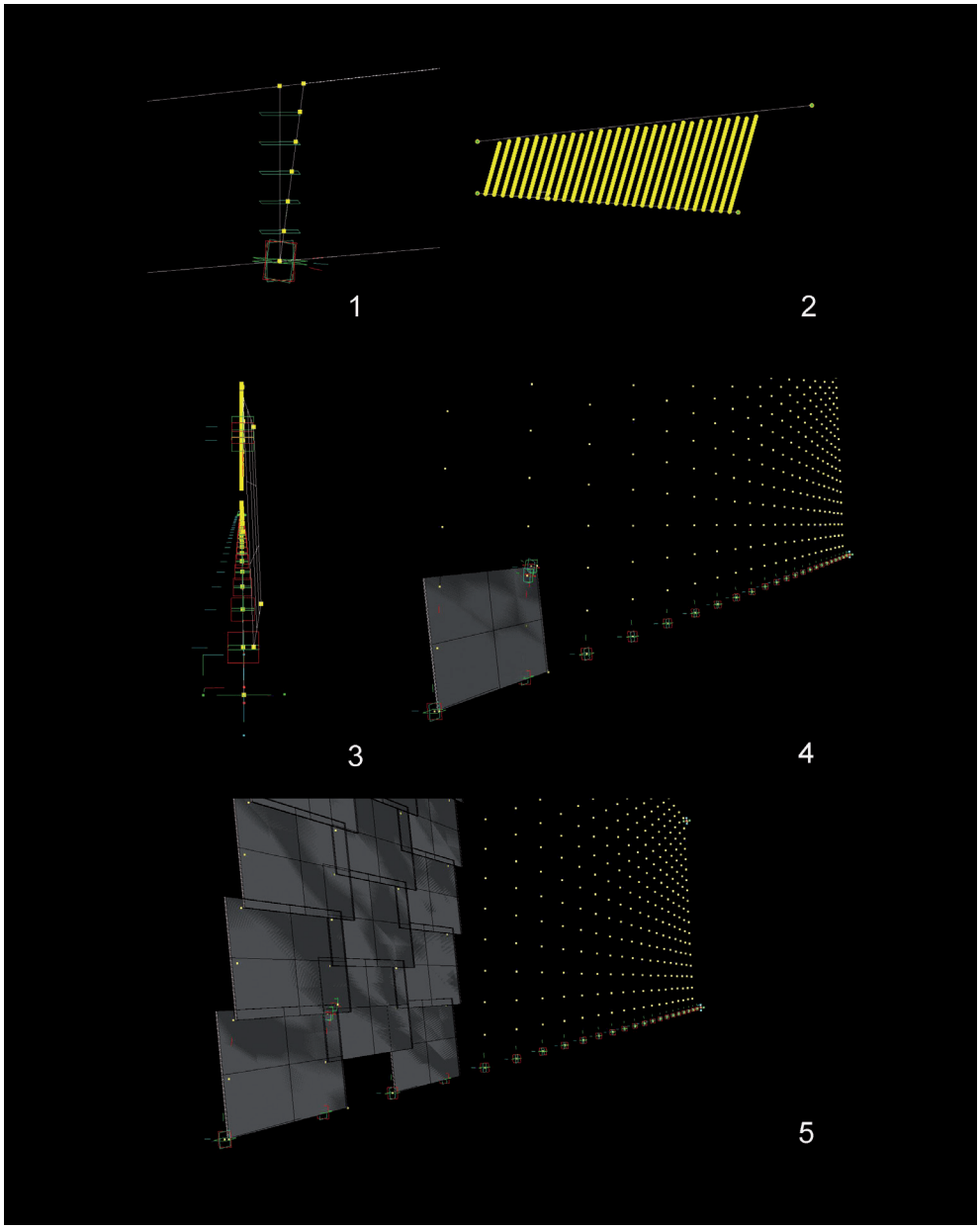


Figure 40
GC components for the fish-scale façade. The sequence to produce the cladding between two arbitrary curves was as follows: component (1), that generated the grid, was placed between two curves (2); when the grid was generated, then the façade panels could be placed onto the grid; a single point and a local coordinate system were the input to position a shingle component in its tilted orientation (3 and 4); multiple same-sized planar shingles were placed in the overlapping fish-scale pattern on the parametric grid (5).

4.3.6. Summary

The largest proportion of the GG group's involvement was in the analysis, the communication of the nature of the geometric problems, and the set-up of an operational framework which encapsulated constraints and flexible parameters for rationalisation.

At the beginning of the collaboration, an ambitious global parametric model was discussed, which was thought would even include the façade tessellation. The project proved a good pilot study, to understand that a fully parameterised model needed a clear description of all parameters which might change and those which would be fixed. The initial set-up of the dependency model was time-intensive and the parameters could not easily be switched from fixed to flexible.

The project showed the limits of GenerativeComponents. It was in its Beta version and unable to handle larger model hierarchies without an expansion of its capacity through writing C# code. Only parts of the structure and façade were modelled in GC, and later transferred to a non-parametric principle model. The experience led to step-by-step customised code development within the group, which was reused in later projects.

In this scheme, a homogenous appearance of the external hull and the internal surface was desired. The layers of the skin, meaning the external façade, the roof built-up (waterproofing and services), and the structure and the internal skin of the exhibition space, were anticipated as regular offsets from each other. This was different from a design where the freeform façade was suspended from a regular orthogonal structure. However, a regular offset would have caused drastic jumps of the internal building skin into the exhibition space, as described before. The visible outer and inner envelope skins appear homogenous, even though they were in fact modelled as two independent surfaces. This could not be avoided, as the character of the building hull, elicited by the rapid changing radius of the ridge angles, had to be maintained. A unification of the angles would have caused the loss of the design's dynamic appearance.

Technically, it might be desirable to start from a central surface definition from which to offset the internal and external layer, thereby providing a greater control of all the necessary layers. In practice, however, for most architects who operate in the arena of complex building hulls, the full control of the external exposed building form seems vital, so the outer surface becomes the starting point.

Continual changes in the internal building programme were extrapolated to necessary changes to all the building layers. This meant that each layer was repeatedly remodelled. In particular, the creation of the structural layer was work-intensive. Curve offsets from the external skin had to be replaced by straight polylines. This was necessary to allow for the structural analysis by Autodesk's structural analysis package 'Robot' and for representing the standardised beam segments. To speed up the design analysis iteration cycle at the beginning of the development, it would be beneficial to employ an analysis package which handled curves. The changes in the geometry were too big to write an automated routine which generated a straight line model from the curves of the design model. Therefore, every time something changed, a new structural line model had to be created. This was the classic dilemma, which is looked at in the current agenda of Building Information Modelling (BIM).

For the design of the façade, the idea was that a regular tessellation with same-sized, flat panels would be cost-effective. However, it was not. The overlapping fish-scale design, in particular, was complex. While manufacturing the same-sized flat shingles from the sheet material would indeed be inexpensive, the time resources which went into the trials for defining the detailing of such a tessellation expanded and finally caused a change in the design for the façade.

When a solution for the initial design had been partially developed, the design of the façade was changed from the overlapping fish-scale solution to a continuous perforated roof. This cladding option was less demanding to model, as it could be defined more easily at the difficult sections of the roof, such as the tight radius of the ridges. Most of the façade's panels had different shapes and when manufactured, had to be handcrafted. At a later stage, Arup Engineers optimised the cuts of the perforated material so it had parallel edges where possible and this eased the cutting

process. It would have been interesting to see if a subdivision following the scaling of the geometry would indeed have been more expensive than the attempts to impose different rationales on the shape.

Placing same-sized and flat elements onto a highly articulated form, as desired at the beginning, was a challenge in its own right and was investigated separately outside of the project context. Both the global geometry and the local tessellation system had to be understood so that common characteristics could be found, to ensure compatibility between the two systems. It was a part of the recurring scheme of different orders being placed on top of each other.

Achievements

- Clarification of geometric problems in the global form.
- Partial rationalisation of key geometry in the global form to ease the geometric development, and the definition of the structural and internal offsets.
- Development of a parametric procedure for producing a constrained fish-scale fragmentation on doubly curved surfaces or between two arbitrary curves.

Relevance for the research project

- Freeform versus complex form.
- One order on top of the other.

4.4. High Court of Justice and Supreme Court

4.4.1. Project description

The project involved the design of the roof of one of two new buildings, designed by Foster + Partners, in the Campus of Justice in Madrid (building on the right in Figure 41). The roof sits on the circular volume which houses the High Court. The GG group created the structural model as an offset from the design model and detail-designed the structural members, paying particular attention to the nodes.

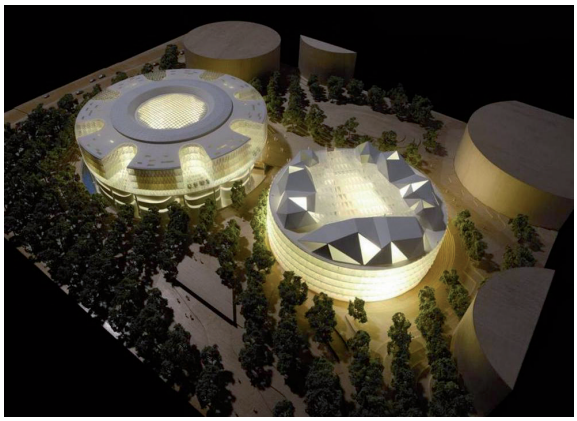


Figure 41
Model of the Campus of Justice in Madrid (Foster + Partners, 2007).

The task was to define the structural members from an external triangulated design surface. In the first attempt to do this, the individual panels were offset by the necessary depth normal to the original surface, to allow for cladding, insulation and services. However, the edges of adjacent panels did not meet, and subsequently their corners did not meet (Figure 42). This was problematic for defining the nodes of the structure. When extending the edges of the normal offset surfaces, the edges of some elements would simply not match up. The overall design had been agreed and signed off, so no changes were possible to the form, except for slight vertical tuning of the nodes. The task for the Generative Geometry group was to find an alternative way to define the structural offset other than using the regular normal offset procedure.

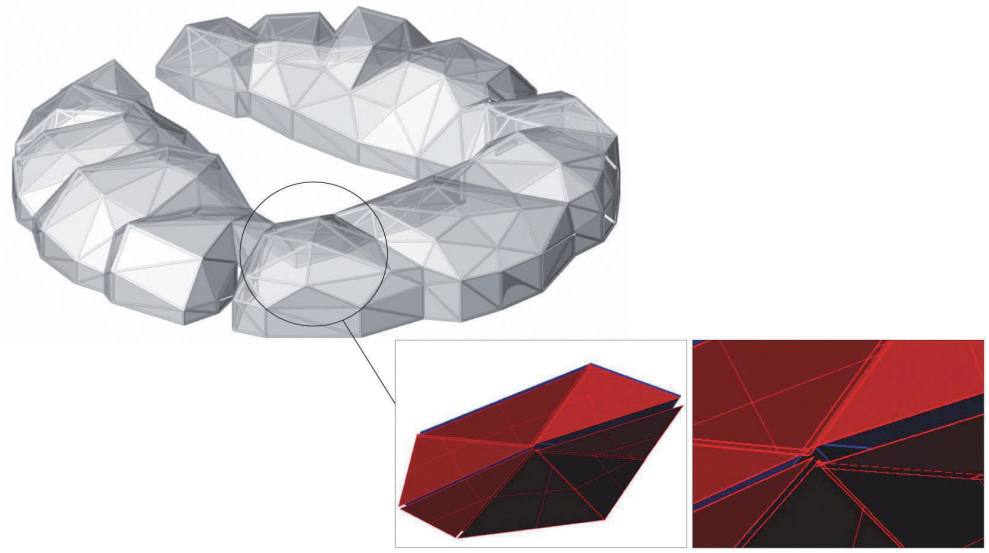


Figure 42
Offset trials. The diagram illustrates the problem which occurred when trying to produce the structural layer by using a regular offsetting procedure. The close-up screen shots show one of the node points where six elements should meet. The blue elements in the background are part of the design surface; the red elements indicate the regular offsets, normal to the initial surfaces. They neither meet in a single node, nor do the edges match.

4.4.2. Process

The solution was a custom script which searches for common node offsets for adjacent surface elements (Figure 43). The script procedure is as follows. The normal vector for each neighbouring surface element is found and they are summed to create the average vector (Figure 44). As the average vector will not be valid for every adjacent panel, given the constraint of the necessary distance to the external shell, a search follows. The script iterates randomly through small changes in direction and scale of the average normal, to find a common vector valid for all adjacent surface elements (Figure 45).

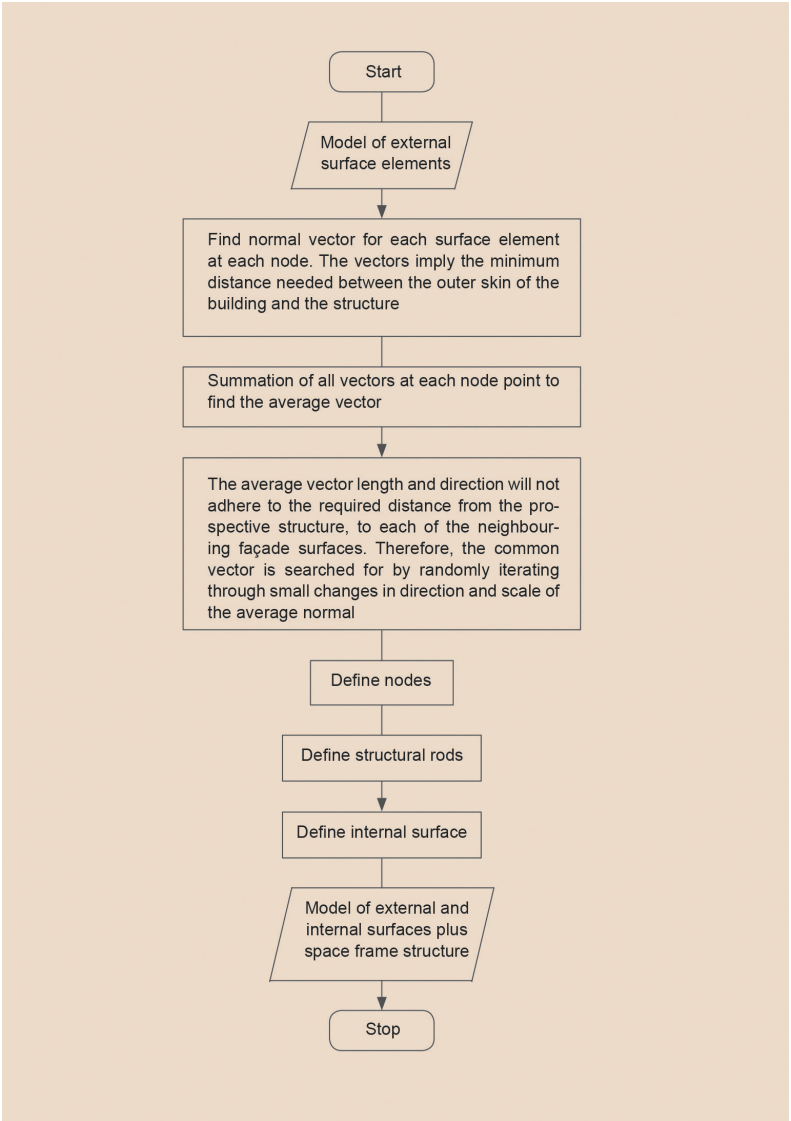


Figure 43
Madrid flowchart. The flowchart describes the search routine to define a common structural offset for the outer folded roof design of the Madrid High Court.

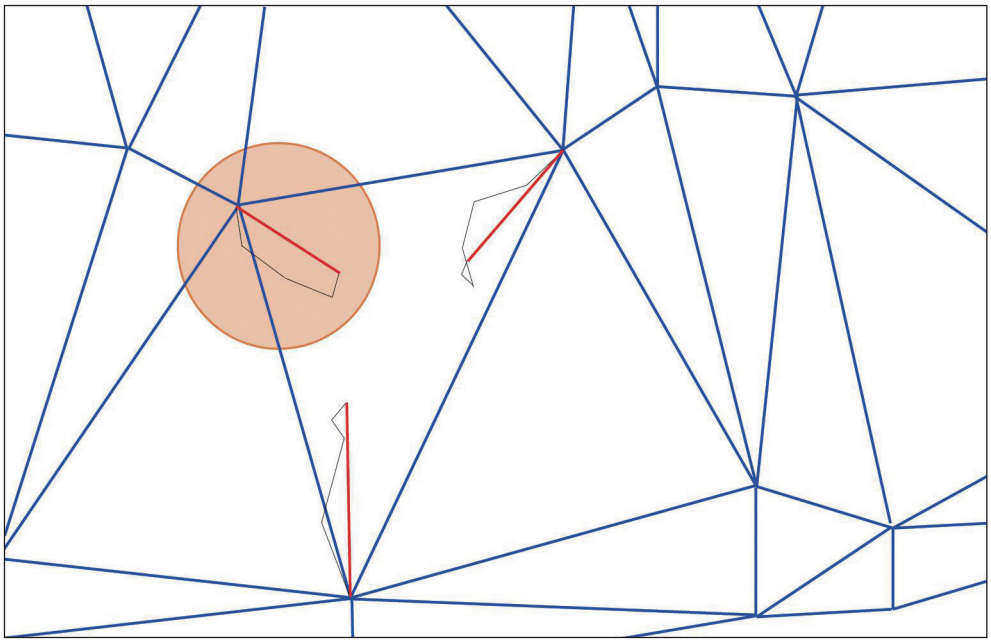


Figure 44
Vector summation. Summation of all normal vectors (black) of each surface (blue outline) to find the average vector (red).

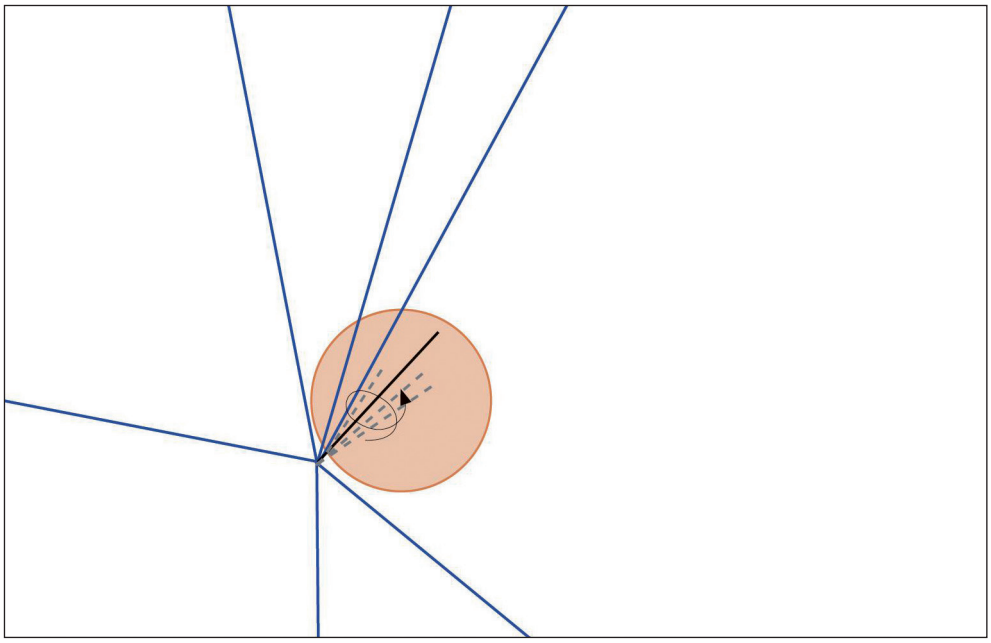


Figure 45
Search for common vector. Searching for the common vector (black) by starting from the average vector, and iterating through small scale and direction changes (dotted lines).

4.4.3. Summary

The process allowed the automation of the definition of structural members and internal cladding as an offset from the design surface. The structure visually mimicked the outer design surface and it kept the necessary distance for the roof build-up, while not taking up more space than necessary to define common nodes between adjacent panels (Figure 46). The problem was somewhat similar to that of the Museum of Transport, where a regular normal offset did not produce a useful structure and internal surface definition. Both designs received internal surfaces that only imitated the outer skins visually. The internal ceiling of the Transport Museum was modelled

manually; the Madrid Law Court project called for an automated procedure, as six surface faces met at every node point and the problem became too big for a manual modelling task. The code to define the offset for the freeform folded surface was reused on the same or similar problems later on. Akin to the Glasgow project, the designers were surprised that it was not possible to create regular offsets. They had not anticipated the amount of time required to solve this problem. In addition, the cost was higher than expected due to all the nodes being different.

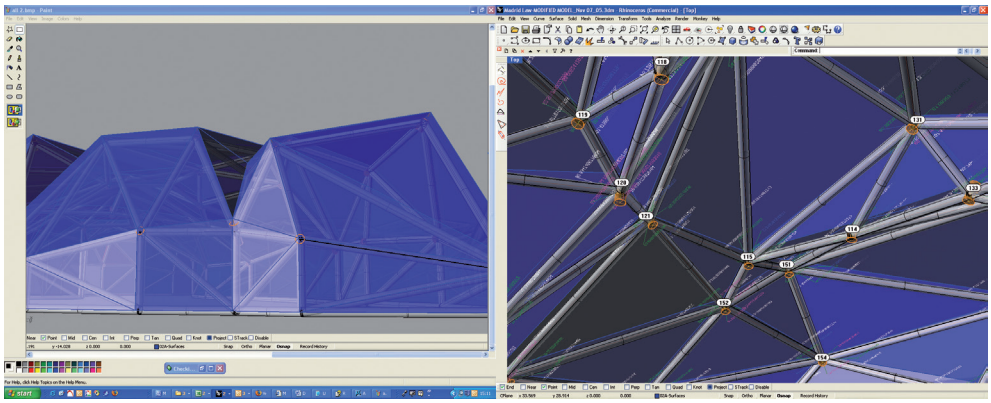


Figure 46
Detailed model. Resulting definition of structure and description of nodes.

Achievements

- Definition of the structural offset and internal cladding from the freeform folded design surface, considering minimum distance constraint from the external envelope.

Relevance for the research project

- The intricacy of defining physical offsets for material, roof build-up, services and/or structure from freeform design surfaces.

4.5. Zagreb Airport

4.5.1. Project description

The project was a competition entry for a new airport terminal in Zagreb, Croatia. The architects were Foster + Partners, and the consulting engineering firm was Buro Happold, with the GG group collaborating on the design conception. The proposal came third in the competition, but it was reconsidered for realisation at a later stage. This was because its modularity made it cost-efficient and easily extendible, while maintaining its visual complexity (Figure 47).

Figure 47
Zagreb Airport. Competition rendering (Foster + Partners, 2008).

4.5.2. Design built-up

In contrast to the other case studies, the competition entry for Zagreb Airport is not only documented here for its interesting problem solving, but also because the design takes on the idea of creating complexity from the assembly of simple parts. Repetition was a crucial criterion, and the assembly building parts were developed in collaboration between the architects and the engineers from early on in the design process. The cleverness of the design lay in its visual complexity, which seemingly contrasted with the high degree of repetition. It was a modular assembly of structural roof modules. The whole design was referenced to a twelve-by-twelve metre grid. The roof built-up consisted of three layers: two two-dimensional grids, each with a different pattern, and a linking three-dimensional bracing structure (Figure 48). The main grid was based on a Penrose tiling, where two types of rhombi were arranged. The second grid layer was a Voronoi diagram. The design anticipated the grid layers to be broken into smaller modules, which were prefabricated and connected on site.

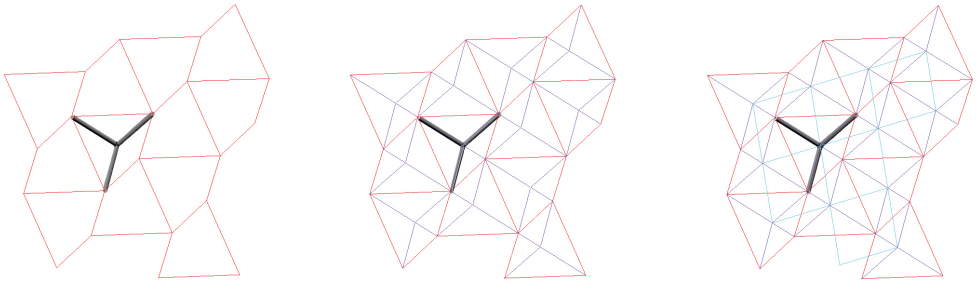


Figure 48
Two grids and bracing. Two two-dimensional grids (red and purple) and three-dimensional bracing between the two grids (turquoise).

The footprint of the airport, as well as the roof design, appeared random and yet was greatly repetitive. Only the opaque cladding panels of the outer layer were placed randomly, which resulted in a play of light resembling a carpet of leaves in a forest (Figure 49). In addition, the columns were placed irregularly, whilst incorporating the internal program of the airport and the structural performance.

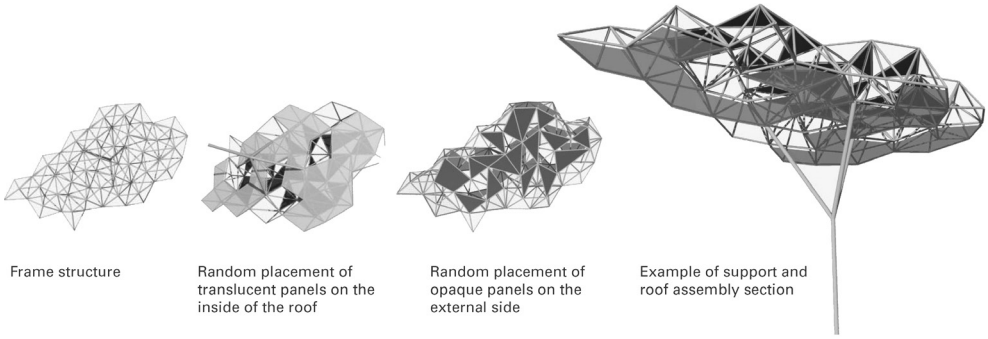


Figure 49
Roof built-up. Complexity was added through the irregular distribution of opaque and transparent cladding panels.

4.5.3. Summary

The Zagreb competition entry was not the only recent design by Foster + Partners to focus on modularity. The advantages of repetition in building parts allow for the expansion of a building over time, cost-efficiency and aesthetic coherence. The Queen Alia International Airport by Foster + Partners is another example where expandability was a crucial design factor. Here, modularity was reduced to a single palm tree-like column-canopy unit.

The observation that aesthetic complexity can be derived by an inductive assembly process or growth of simple repetitive parts has been important for this thesis. Even if the parts are manually assembled, the crucial point is that the accumulation of parts corresponds to certain spatial arrangements that are allowed to change and develop through the design progression, as opposed to a predefined definition of global form. The parts inherit information on materiality, manufacturing and construction. The number and arrangement of the parts does not overly affect any of these considerations.

Achievements

- Definition of modular roof detail which can be easily manufactured, constructed and extended, whilst creating a visually intriguing design.

Relevance for the research project

- Simple repetitive parts, informed by physical parameters, are put together to form a grander, visually complex assembly.

4.6. New Holland Island

4.6.1. Project description

Foster + Partners' New Holland Island project comprises a roof structure spanning the courtyard of an old circular building complex, formerly a prison (Figure 50). The objective for the roof was to cover the space whilst allowing as much light through as possible. A steel-glass structure was proposed. Buro Happold engineered this roof for the planning application.

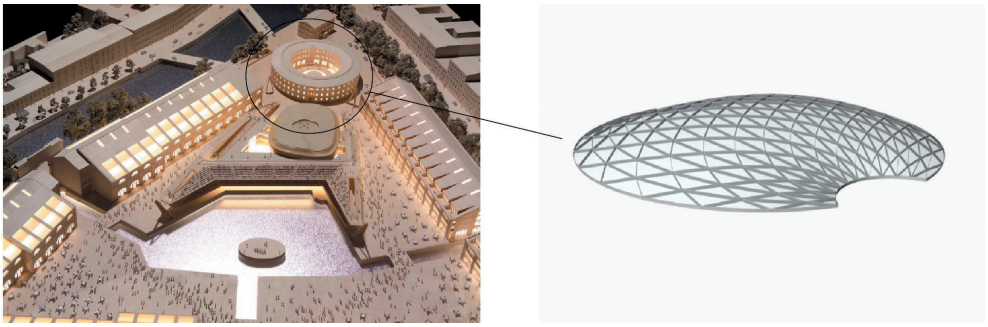
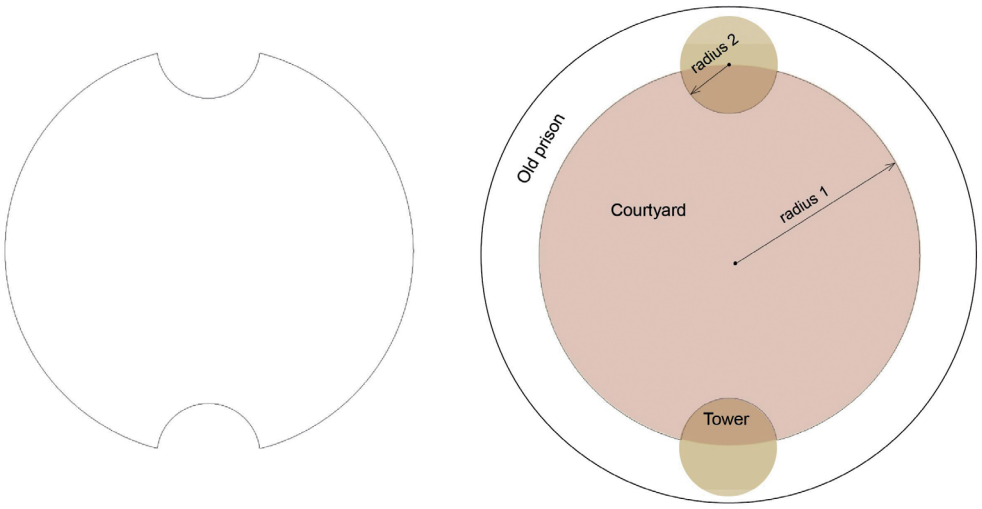


Figure 50
New Holland Island. Master plan model of the project (left) and initial digital design for the roof (Foster + Partners, 2006)

4.6.2. Project brief

Two towers in the old building structure demanded two round cut-outs in the circular edge of the roof, as shown in Figure 51. These cut-outs added complexity to an otherwise simple circular footprint, which on its own would suggest a pure dome formation. Due to the cut-outs, however, the form had to accommodate the changes in curvature of the outline, to avoid stress accumulation. It was only possible to make assumptions about the structural performance of the existing building, therefore it was important to limit the amount of introduced lateral force. To do this, it was necessary to have an even flow of forces.



Outline for the prospective roof structure

Diagram in plan on the Rotunda site

Figure 51
Roof outline. Diagram of the rotunda roof diameter, showing the two round cut-outs for the towers.

The GG group was approached to help resolve a problem regarding the sizing of the structural rib sections. The group was asked to investigate whether the structural performance of the overall geometry could be improved to allow for smaller rib sections. There were two arguments for trying to achieve a more structurally efficient roof: a) increasing the transparency and therefore high natural light penetration and b) reducing the material cost. Fewer ribs and slimmer rib sections would serve both objectives. The improvement of the structural performance would therefore fulfil the architect's aspiration. However, there was the question of how far the geometry could be altered to help achieve this goal.

Engineering the project implied studying the underlying geometric framework, and how the design could be informed by structural performance and manufacturing

processes while meeting the expectations of the design. The efficiency of the structure was determined by three main parameters:

- curvature of the roof
- local geometry and global layout of the grid elements
- employed material.

Materiality, the principal layout pattern of the grid and the details of the rib sections were defined from the start of the GG group's involvement. The roof was thought of as a steel-glass construction. Foster + Partners wanted a diagrid as the main layout for the roof beams and had a preference for triangular cross sections, to give a lighter appearance to the structure. In the ongoing process, the height of the pitch was a constraint too, but the overall form of the roof remained flexible and open for optimisation. The flexible parameters were therefore limited to the number of ribs, determining their density and distribution in the layout.

4.6.3. Measures for comparative analysis

Through project work and working closely with engineers, the author gained insight into the process of analysing architectural designs. This applied to comparisons between different design alternatives at the beginning of the design route and at a later stage during the engineering (sizing the structure). In this research, the focus is on the initial evaluation and the comparison between alternative designs. Strength and stability analyses are the main factors when structuring a design. The importance of strength and stability in the course of analysing a structure depends on the span between the supports. The wider the distance between the points where the forces are transferred into the ground, the more important it is to know what the deflections would be, to determine the strength of the structure. The closer that the supports are placed to each other, the greater the significance of the stability analysis. Experience proved that at a span of around 10 metres, both performances needed to be examined equally.

Dr. Andrew Chan is a senior engineer at Buro Happold, who has been responsible for the engineering of many important complex structures. The author talked to him about what the comparative structural performance measures should be in an optimisation cycle that was part of a digital form generation and form optimisation process. He said that for a short-spanning structure (roughly under 10 metres), measuring the stress is important, and for a long spanning structure (more than 10 metres), measuring the deformation is important.

In a digital generation and optimisation process the distance of the next generated structure is unknown; one cannot predict whether the maximum span will be under or over 10 metres. As a result, one cannot know if stability or strength is more important. The system therefore has to handle both short and long distances between supports. Chan decided that in a consistent comparison, even at an early stage of the design (the initial design comparison), stress and deformation or strain should be measured. He concluded that if only one measurement is implemented in the process it cannot be a valid structural judgement. Figure 52 shows Chan's sketch of the relationship between relevant forces and the span of a structure.

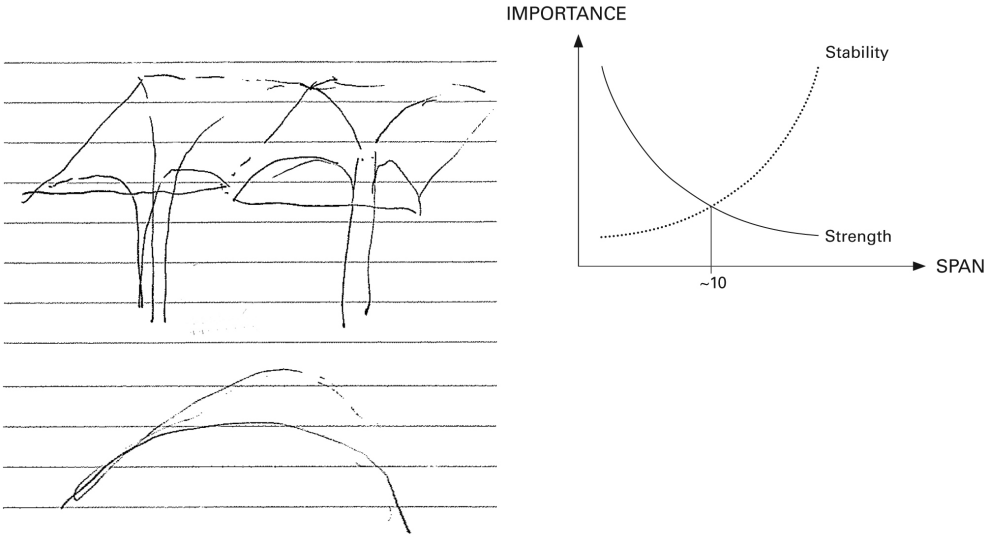


Figure 52
Measure for structural performance comparison. Structures with shorter spans between supports require attention to the stress (top left). Structures with a wider span require attention to the deformation (bottom left).This is illustrated in the diagram (right).

4.6.4. Process

In the following section, the process steps are listed which lead to the final roof grid layout and overall form of the roof. Thereafter, the steps are described in greater detail.

Grid

- Comparison of three grids: the diagrid on its own and two different versions with a secondary grid.
- Stress diagram illustrating the force distribution according to boundary conditions.
- Form-finding, allowing for shifting and rotation.

Global geometry

- Analysis of the initial geometry.
- Optimisation of the initial geometry.
- Form-finding the geometry.
- Analysis of the form-found geometry.
- Scaling while form-finding.
- Comparison of the initial and the form-found geometries.
- Decision on principal model.

Defining the grid layout of the roof

First, the grid layout was tested, taking the architect's model and grid as the reference. Figure 53 shows three different grid layouts with their exaggerated displacement diagram. It shows how a change in the organisation of the grid affects its structural behaviour.

The diagrid was compared to two directional variations of the secondary grid. All three versions had the same specifications of rib profile, size and material properties, to allow the comparison.

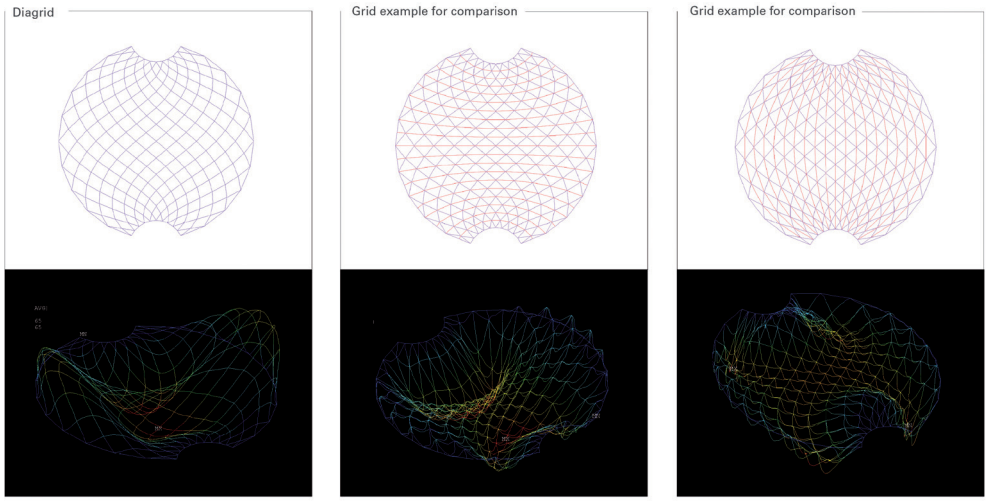


Figure 53
Grid comparison. Scaled deflection diagrams on the diagrid. The layout of the diagrid and its deflection diagram (left). The diagrid with two variations for the secondary bracing and the corresponding deflection diagrams (middle and right).

The diagrid could structurally support itself and, in order to do so, it needed a denser grid than the initial model suggested by the architect. Its layout was mainly defined by the available glass panel sizes. In addition, the structural members needed to be more substantial. As in the case of a pure diagrid, the grid subdivision was not triangulated, and the glass panels needed to be curved or, if planar elements were used, the surface had to be stepped. Stepping had an interesting visual effect but was difficult to keep clean (Figure 54). Curved glass panels were not considered as an option, because they were too expensive.

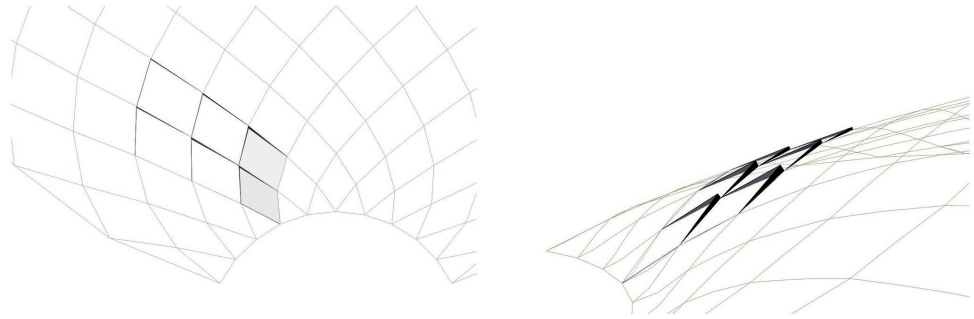


Figure 54
Stepped glass panels. In the case of a diagrid alone, with no secondary bracing, the glazing would be four-sided. In order to avoid doubly curved panels, the planar glass panels would have to be stepped.

Bracing the diagrid with a secondary grid had two advantages: first, the individual members could be slimmer; second, triangulated planar glass panels could be installed. As the secondary grid needed to support the load-bearing of the diagrid, the direction of its members needed to be informed by the force flow. Assessing the second and the third grid options, which showed the bracing, the members in the second variation were shorter overall and thus structurally more efficient. The third variant exhibited efficient members spanning between the two towers, but toward the outer areas of the form, the ribs spanned a long distance. Therefore, these ribs did not contribute to the structural performance and, on the contrary, added more weight to the roof. Nevertheless, the architects preferred this variation, because the structure appeared less constrained and thus more open. In the structurally more

efficient version (option 2), the bracing made the structure look stiffer, but it would have allowed for less material in the individual load-bearing members. It was decided that the system that would be employed combined triangulated members which work structurally in some parts of the roof, with members in other roof areas that were jointed so that they did not attract any loads.

The architects were interested in further exploring the idea of a structurally ideal grid layout, in relation to the load impact and support constraints. Winslow, together with Buro Happold and Cambridge University (Peter Winslow 2009), looked at force patterns on complex-shaped forms to propose structural member layouts. The study allowed an assessment of what the ideal grid layout would look like with the given constraints for the New Holland Island rotunda roof (Figure 55). The diagram shows that the force pattern is more complex than the appearance of the dome-like design suggests, solely because of the cut-outs in the diameter.

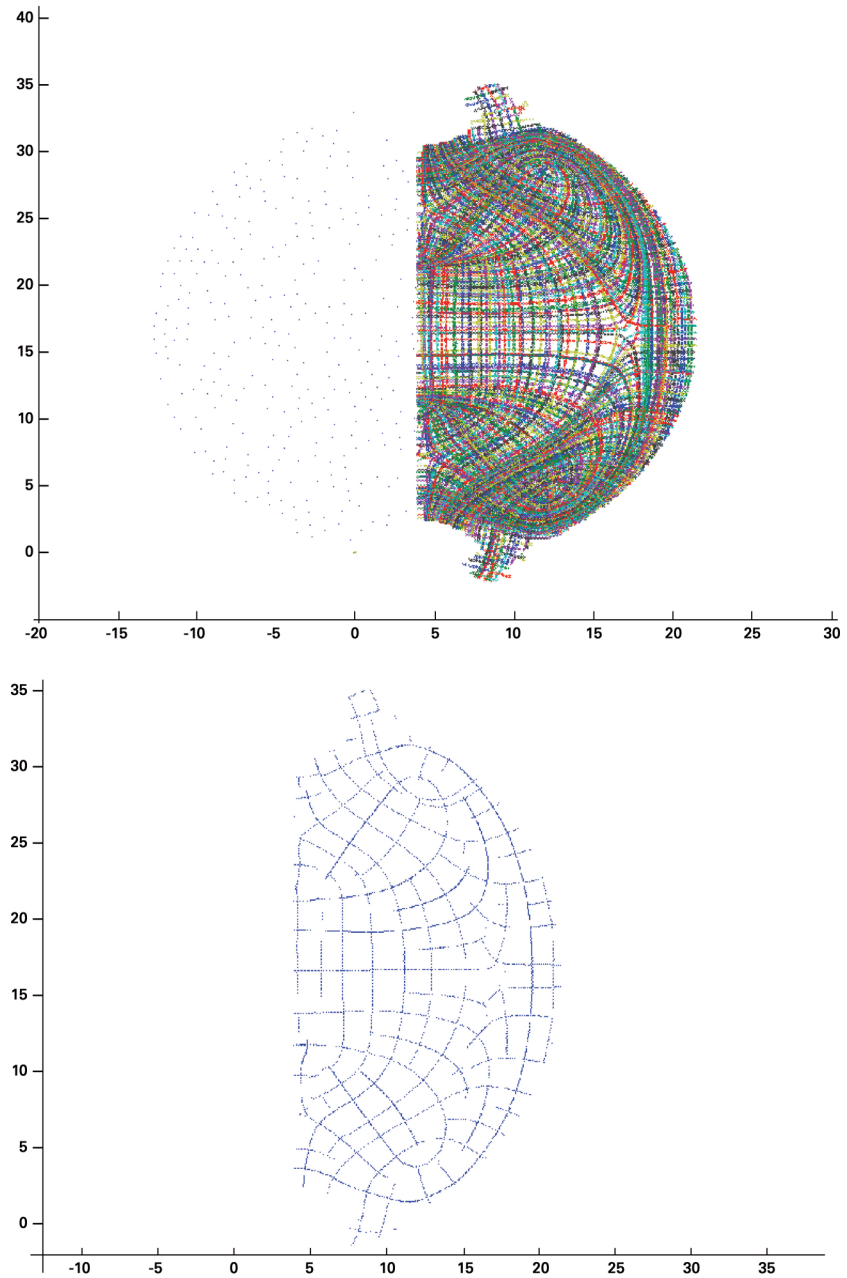


Figure 55
Stress diagram. Stress pattern under dead load (top) and (bottom) the same stress pattern with a 2m spacing constraint (Peter Winslow 2009).

Definition of global geometry

In the following section, the form-finding of the global structure is described, including allowing the grid members to shift and rotate whilst relaxing under inverse gravity. Freeing the rotational movement during the form-finding process resulted in an obvious change in the member layout (Figure 56). The ribs ran straighter into the ring beam. They also clustered where there were high curvature changes around the towers and spread where there was a regular dome-like articulation at the sides. What made sense, in terms of structural performance, conflicted with the distribution of light into the inner building, as well as the maximum achievable size of the glass panels. It was, however, advisable to follow the straightening of the ribs in a controlled way.

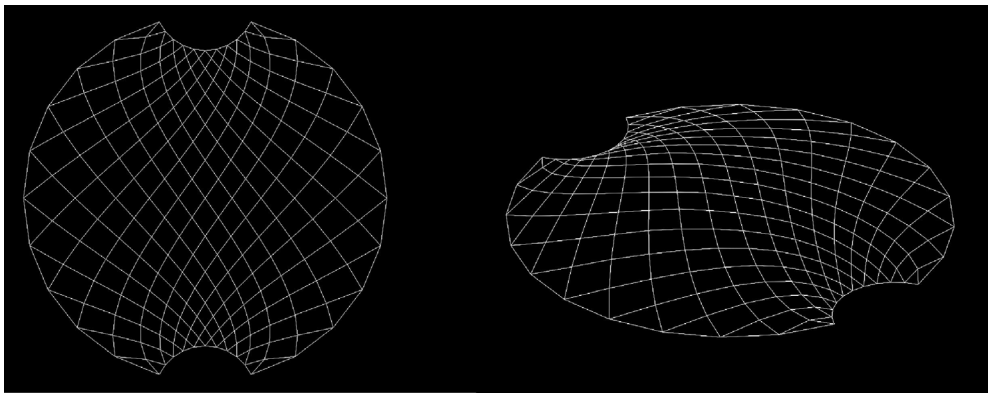


Figure 56
Freedom to yield in all directions. Grid layout when allowing for rotation and shift during the relaxation process.

Initial geometry

The initial geometry by the architect, with a mid-span height of 5 metres, was analysed using the diagrid as the fixed parameter. The material and size specifications for the comparative analysis implied relatively slender sections and a larger ring beam. The structure was analysed simulating self-weight and snow loading. The overall directional deformation under self-loading plus stress distribution was extracted; see the upper row in Figure 59.

Form-finding global roof geometry

The geometry was generated through an iterative relaxation process. The material rose under its own weight in an inverse relaxation process, from a planar grid layout into a three-dimensional form (Figure 57). This procedure followed the same concept as a physical form-finding process, such as a hanging chain model or soap film experiment. Every framework of the physical specification resulted in a single configuration as an ideal distribution of forces and accommodation of geometric edge conditions. Here, the edge condition was the layout of the grid and the circle with two cut-outs, which reversed the continuing curvature of the circle. An alternative to generating the geometry from scratch was to take the geometry given by the architect and allow it to relax into shape. Both approaches were investigated.

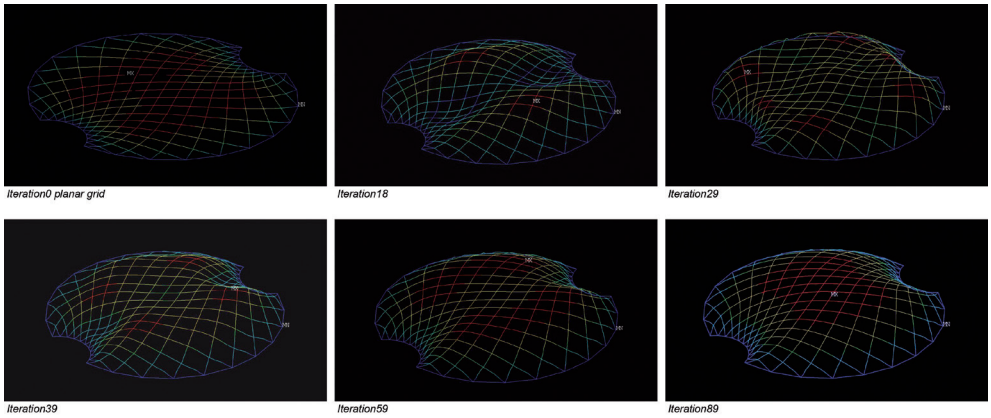


Figure 57
Digital form-finding.

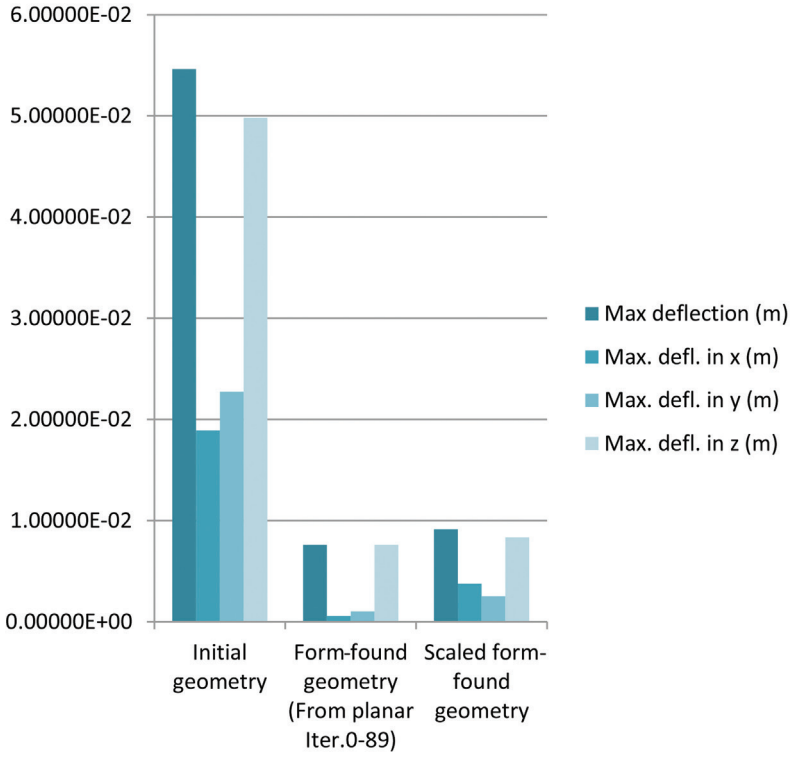
One hundred iterations were performed in the form-finding process from a planar grid to the 3D form. The specifications for the analysis were the same as those for the initial geometry, to allow the comparison. The resulting pitch height was only 2.843m. However, the performance was better than that of the initial design, despite its lower rise, because of its optimised curvature distribution. The analysis diagrams in Figure 59 indicate the stress distributions in red. The optimised version shows a different stress distribution (see the middle row of Figure 59), to the initial geometry (upper row of Figure 59). The overall deflection and the von Mises stress were significantly reduced. The resulting form had 86% less deformation and 42% less stress than the initial form.

Form-finding geometry with scaling factor

The initial geometry provided by the architect had a pitch height of 5 metres; in comparison, the generated geometry had a pitch of 2.8 metres. The height of the evolved form was due to the geometric constraints, as well as the physical specifications, namely the definition of the material and the sizing of the section.

Changing the physical specifications of the simulation, by reducing the size of the structural elements, achieves a higher pitch, but the height cannot be controlled. Theoretically, a relative scaling of the geometry was possible by quantifying the form alterations during the form-finding process. The scaling created a form with a controlled pitch height.

In doing so, however, an assumption was made that a linear scaling process could be utilised on a non-linear form-finding process. In areas such as the transition between the two diameters, where extreme curvature changes occurred, the scaling did not do the same as the form-finding process; the higher the roof, the greater the geometric changes in those areas and the higher the stresses. Therefore, it was essential to simulate the structural behaviour again after scaling it (bottom row in Figure 59). However, as Figure 58 shows, the deformation in both geometries, i.e. not scaled and scaled, was kept to a minimum (86% less maximal deformation than in the initial geometry) and the higher roof even improved the overall stresses (66% less total stress than in the initial form). While maintaining a similar curvature definition to the pure form-found articulation, the scaled roof form also exhibited a stiffening effect through the higher mid-span height.



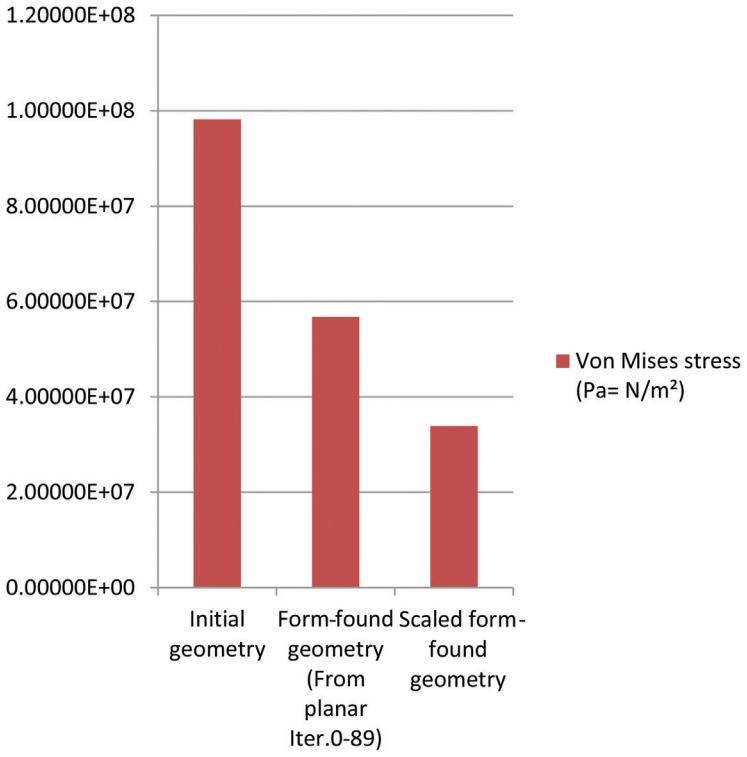


Figure 58
Structural performance comparison. The two charts compare the displacement (top) and stress reaction (bottom) of three global geometries: the initial geometry, the form-found geometry and the scaled geometry.

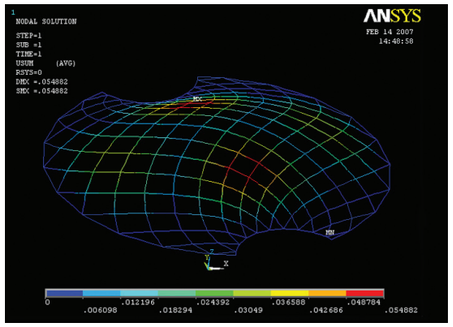
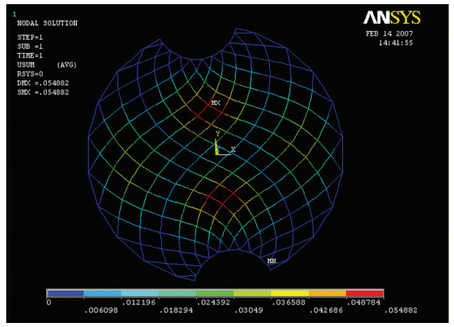
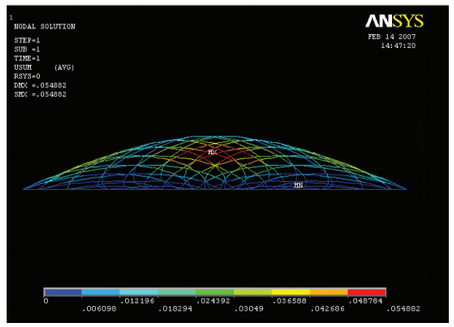
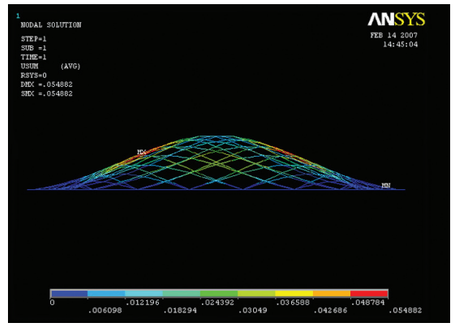
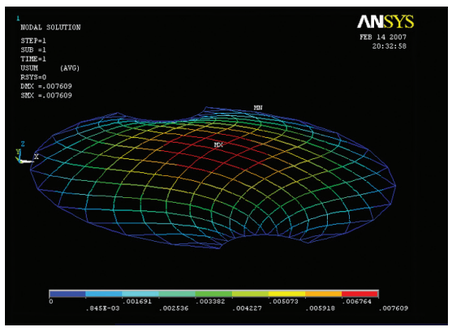
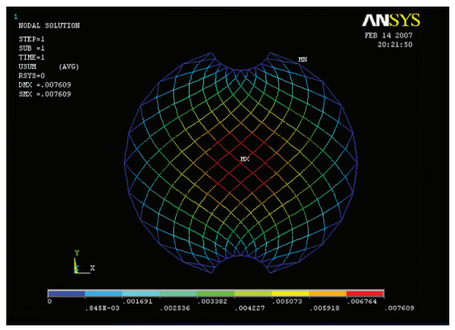
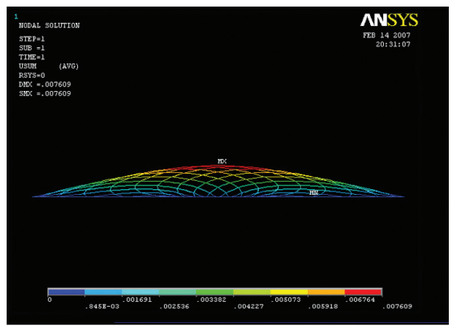
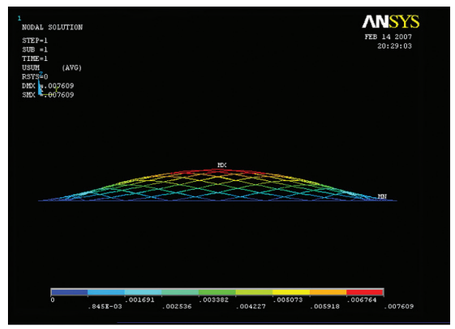
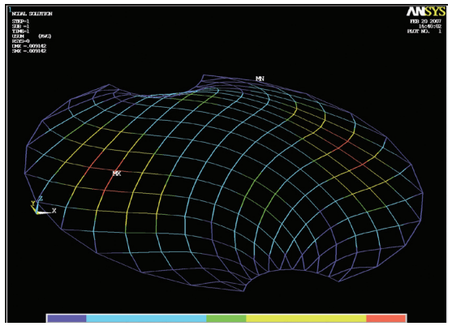
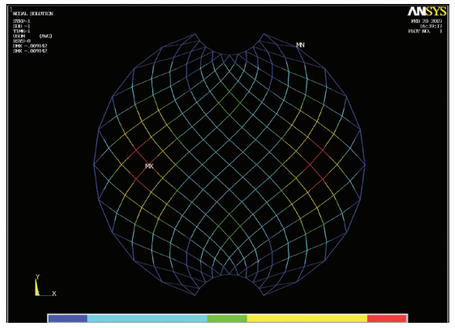
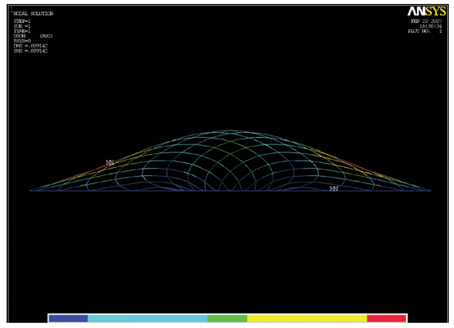
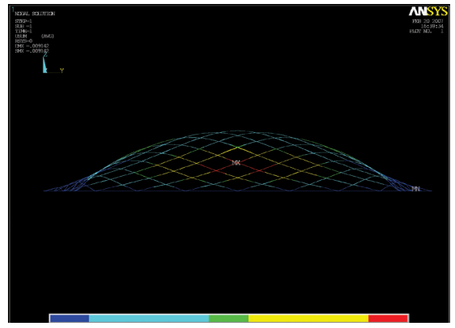
	Isometric view	Plan view	South-North view	West-East view
Initial geometry				
Form-found geometry				
Scaled form-found geometry				

Figure 59
Optimisation cycle. Analysis of the initial geometry (top), analysis of the geometry form-found from a planar diagrid (middle) and analysis of the final scaled form-found geometry.

			Material model 1 (general steel beam)		Material model 2 (steel ring beam)		Set up for geometry generation		Set-up for analysis		Output				
Name	**General section (m)**	**Ring beam section (m)**	**Density (kg/m³)**	**Linear isotropic**	**Density (kg/m³)**	**Linear isotropic**	**Supports**	**Load m/s**	**Supports**	**Load m/s**	**Max deflection (m)**	**Max. defl. in x (m)**	**Max. defl. in y (m)**	**Max. defl. in z (m)**	**Von Mises stress (Pa= N/m²)**
Initial geometry	0.05,0.1,0.02	0.5,0.50.05	19720 (steel, glass snow)	EX:2.iE+011 PRXY:0.3	8.00000E+03	EX:2.iE+011 PRXY:0.3	/	/	Fixed in shifting, rotation allowed	9.8 (gravity)	5.46200E-02	1.89000E-02	2.27400E-02	4.98200E-02	9.82007E+07
Form-found geometry (From planar Iter.0-89)	"	"	"	"	"	"	Fixed in shifting, rotation allowed	-4.00000E+00	"	"	7.60000E-03	5.73000E-04	1.02000E-03	7.61000E-03	5.68000E+07
Scaled form found geometry	"	"	"	"	"	"	/	/	"	"	9.14000E-03	3.76000E-03	2.52000E-03	8.34000E-03	3.39000E+07

4.6.5. Summary

The main objective was to ensure the roof had a high degree of transparency. This was done by aiming for a lightweight structure. The rib section had to be as slender as possible and the grid layout as open as possible. The optimisation of the overall geometry allowed for an average improvement in the structural performance, resulting in high savings on material, low cutting cost and high transparency of the structure. This proposed roof had 14% of the maximum deformation of the initial geometry and 66% less stress. The proposal thereby produced an efficient form, which met the architect's design intent (Figure 60). The outcome was an optimised global geometry for a specific roof outline and grid layout. The process could rapidly accommodate changes in input data, such as perimeter changes, load or support changes, and alterations in the grid layout.

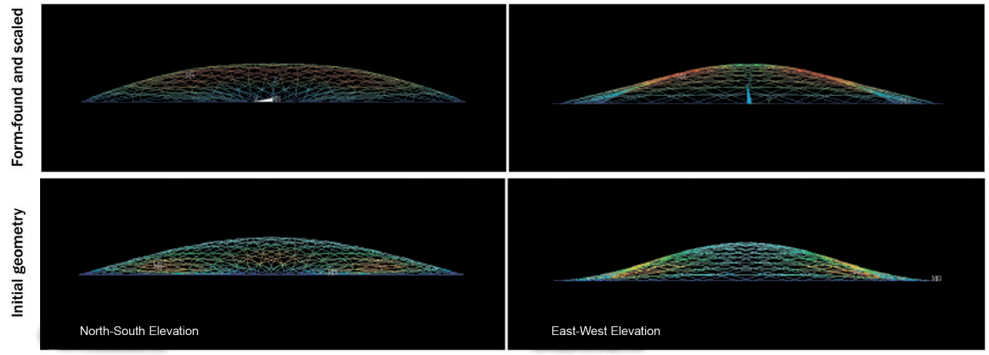


Figure 60
Comparison of the geometry. Geometry elevations North-South and East-West views of initial geometry (bottom) and form-found geometry (top).

It is important to note that the analysis was for comparative purposes only and did not entail structural sizing. Despite this, the finite element analysis that was carried out later on was accurate and the members were kept more slender. This was in order to trigger a clear reaction to the forces acting on the structure.

The geometric model was later supplemented by the secondary grid, to then perform calculations and tests for sizing the structure.

In this research, it was important to explore how the improvement in the structural performance became an argument for the architect to reach design decisions. Engineering considerations could help narrow the choice of many possible design options.

Achievements

- Optimisation of the roof, taking into consideration the minimal lateral forces that could be introduced to the existing support structure.
- Optimisation of the global geometry to allow for smaller sections, higher transparency and a weight-reduced structure.

Relevance for the research project

- Learning about the correlation of form, material and structural performance.
- Improving aesthetic criteria by improving structural efficiency.
- Narrowing design choice.
- Definition of valid comparative structural performance criteria, which can be exploited at an early design stage to compare design alternatives.

4.7. Bergen National Academy of Art and Design

4.7.1. Project description

The Bergen National Academy of Art and Design project in Norway was designed by Snøhetta Architects. The design was a steel-glass roof which spanned from the entrance area and the main court of the Academy over and onto a building complex, without any additional column support. The open air areas covered by the roof were envisaged as additional spaces for the students to work in, which would be sheltered from rain, wind and snow. The shape of the roof had to reflect the artistic activities underneath it. The aim was that the roof would be material-efficient, so that it would have a lightweight appearance, high transparency and would be cost-effective (Figure 61). Buro Happold's GG group was asked to aid the design and structural development of the roof with these objectives in mind.

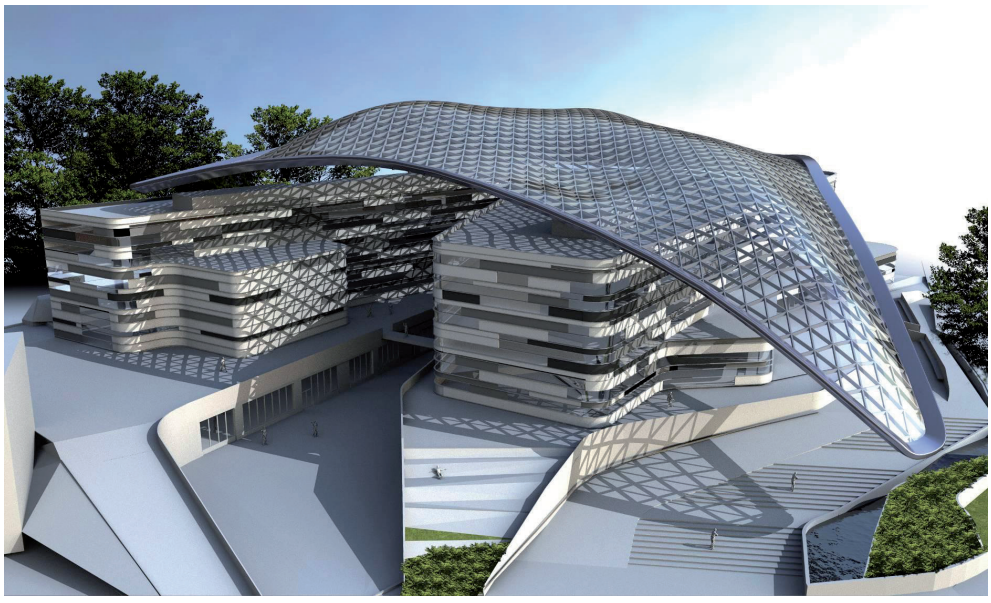


Figure 61
Bergen National Academy of Art and Design. The rendering shows the final optimised roof form, with a continuously triangulated grid (Snøhetta Architects, 2006).

4.7.2. Project brief

There were two main objectives for the project. The first was that the roof had to work as a self-supporting structural surface with no load-bearing columns. The surface freely spanned between the ground and the building complex. The second criterion

was to achieve a wide grid layout of the steel beams, as well as slender and shallow beam sizes, to allow for high transparency.

Both criteria demanded optimum structural performance, determined by the articulation of the roof. During an initial design workshop, the architects learnt that the efficiency of the structure was dependent on the geometry of the roof in relation to its shape, as well as the location of the supports and their characteristics. The architects provided an articulated roof design, hoping that the integration of the dome-like risings would provide self-stiffening. However, due to the support conditions and the complexity of the perimeter shape, the force flow in the roof was complex and the ideal form in relation to its structural performance could not be predicted without analysis.

4.7.3. Process

The GG group developed a relaxation routine which allowed the roof configurations to be form-found in response to specific support conditions. The loads were transferred in an efficient way and therefore the desire to minimise the usage of material could be fulfilled. In order to enhance cost-efficiency, the tessellation of the grid was informed by the warping factor tolerances to produce four-sided elements where possible, thereby reducing triangulation. This also reduced the number of mullions required and thus the number of steel members, it increased transparency and it added an interesting pattern to the roof.

Support schemes

During the conceptual stage of the design, the project went through substantial changes in its overall size and in the locations of supports, which resulted in two essentially different proposals for the form. As the optimisation process was automated, it was possible to adapt the form of the roof relatively quickly to these changing conditions.

The two different support schemes for the roof were informed by changes in the programme and the structure of the buildings that it covered, from which it was partially supported (Figure 62).

Figure 62 Support schemes. Diagrammatic footprints of the roof and indications of the first support scheme (left) and the second support scheme (right).

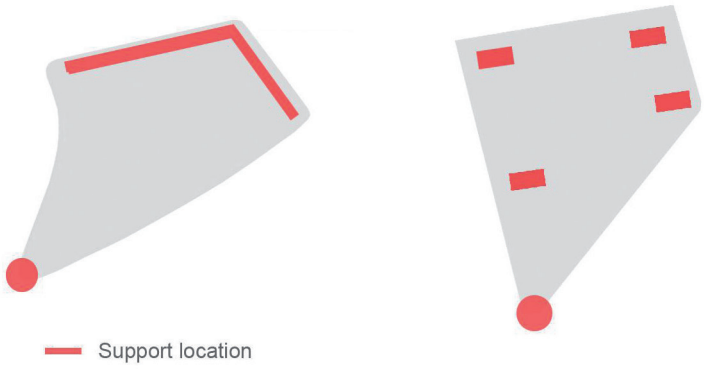


Figure 62
Support schemes. Diagrammatic footprints of the roof and indications of the first support scheme (left) and the second support scheme (right).

In the first support scheme, the roof was carried along two edges formed by the U-shaped building complex underneath, as well as a point support where the structure reached down to the ground. A supporting beam was designed to run north to south, connecting the building and the ground. The footprint of the roof comprises two straight shorter edges forming a field and two longer curved edges leading to a point. The maximum span of the roof was around 125m and its area was around 8400m2.

The starting point for the form-finding process was different in both of the support scenarios. In the first scenario, a new surface was defined spanning the shortest distance within the perimeter boundaries. On this surface, a grid was specified using the UV surface coordinates.

In the second support scheme, a grid was defined on an already articulated surface provided by the architect, and the procedure applied. The architect's roof model exhibited dome-like risings to yield the buildings underneath and to provide what the architect believed to be a stiffening effect to the surface. For the analysis, the geometry in all of the optimisation cycles had the following specification in common to allow for comparison. The geometry was tessellated into a 2m-spaced triangular grid, indicating a network of beams. It was considered to be built with a single layer of steel. Self-weight (meaning the steel under gravity) was applied to the structure, as were additional load forces for glass and snow. An edge beam was assumed, which worked to restrain the roof horizontally and transfer the load into the building. The supports were defined as pinned supports.

Digital form-finding

The form-finding procedure used for the Bergen Arts Academy was the same as that used in the New Holland Island project. It was a custom iterative procedure written for ANSYS, a finite element analysis software application. The deflection was applied in an inverse form. That meant that the initial model was updated and the performance was simulated again. Step-by-step, the form of the roof developed into what could be described as an inverse relaxed topology. The process was repeated until the optimal shape was reached, in which a minimum deflection state and low stress were both attained.

The process allowed the best suitable configuration to be found under the given physical constraints. The advantage of allowing the design to be informed by structural forces was that a structurally efficient geometry directly translated into using less material than one which was not informed by the force flow. This made it cost-efficient in its use of material. It is important to note that there was a difference between the comparative analysis and the final sizing of the roof. The specifications to describe the roof served the development of a shape design and also served to achieve a comparison to another shape with the same physical specifications. The percentage of performance improvement between the alternative forms was proven to be maintained when the actual engineering sizes were calculated.

In the following paragraph, the optimisation cycles for both support schemes are outlined.

Optimisation sequence for first support scheme

- **Original shape** Analysis of the architect's roof form with the first boundary condition.

- **Form-finding cycle** On the architect's first roof boundary scheme, with no given model of the roof form. The roof form was generated from scratch within the boundary, creating a minimum distance surface (Figure 63).

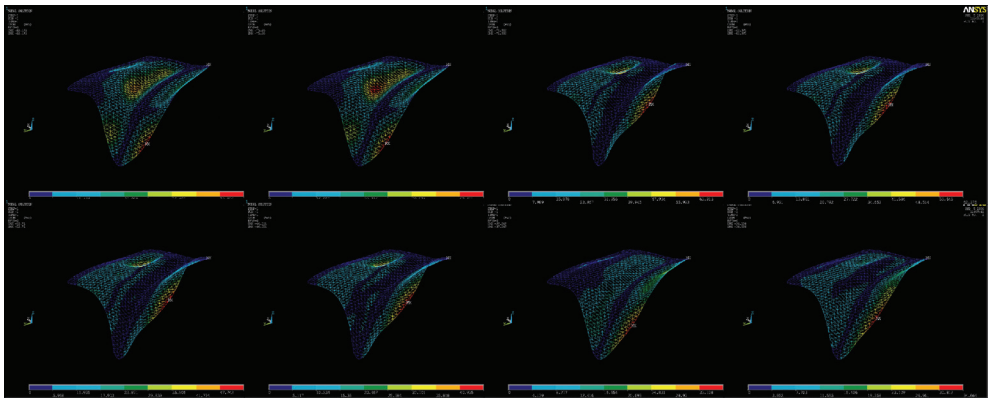


Figure 63
Sample form-finding iteration: 1st scheme.

Optimisation sequence for second support scheme

- **Original shape** Analysis of the architect's roof form with the second support scheme.
- **1st optimisation cycle** Form-finding cycle on the architect's roof form, with the second support scheme and following analysis.
- **2nd optimisation cycle** Form-finding cycle on the architect's roof form, with the second support scheme, using a slimmer section and following analysis.
- **3rd optimisation cycle** Form-finding cycle on the form resulting from the 2nd optimisation cycle and following analysis (Figure 64).

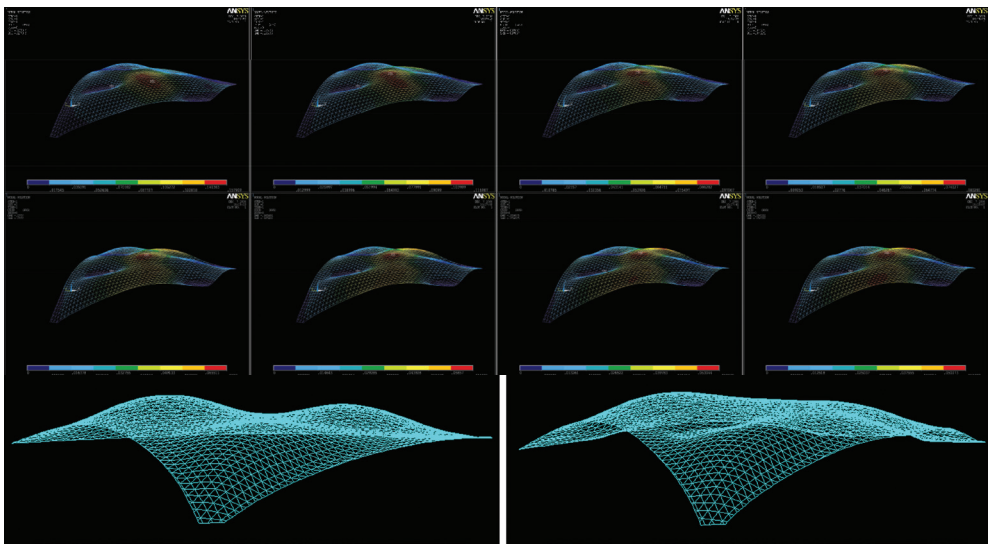


Figure 64
The 3rd optimisation cycle. It starts with the resultant form of the 2nd optimisation cycle. The sequence goes from the top left corner to the bottom right corner. The turquoise line models at the bottom of the capture show the architect's original shape on the left and the form resulting from the optimisation on the right.

Comparative structural performance analysis versus structural sizing

The line model derived from the third optimisation cycle was handed over to the engineers and the architects as the principal model. The architects were not supplied

with the section sizes used for the optimisation, as they only served as comparative sizes. In addition, relatively slim beam sections were chosen, in order to provoke a strong reaction to the forces. The comparative analysis only simulated the effects of vertical loading. For sizing the structure, further load cases had to be tested. To manage the architect's expectations of the final sizing, only a line model and the results of the optimisation were handed over. The structural element sizing followed later.

However, the material savings were reflected in the final sizing of the members. The architects appreciated the resultant complexity in the form, and were surprised at the savings in material and subsequent reduction in cost (Figure 65).

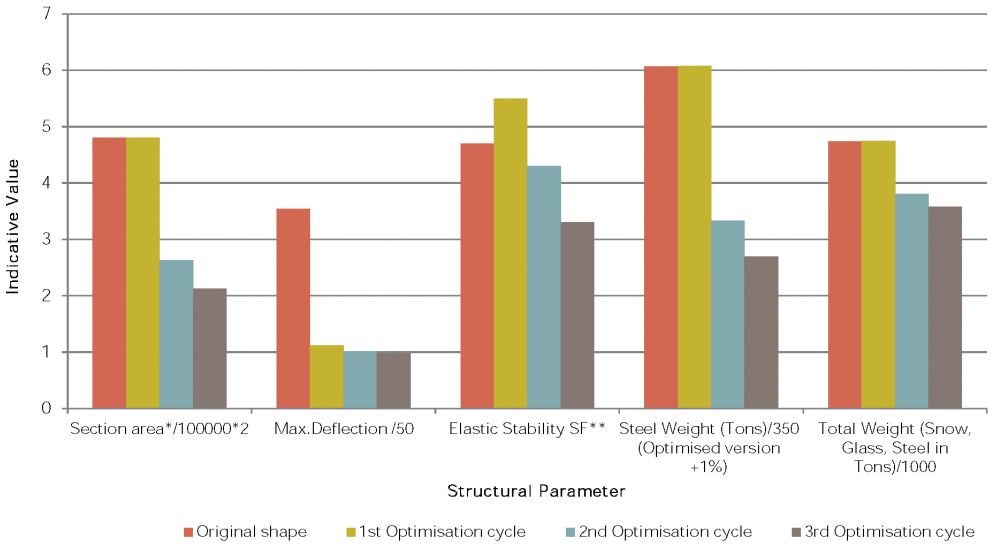


Figure 65
Final comparison. Chart comparing the performance of the original roof form with the outcomes of the different optimisation cycles.

Grid tessellation exercise

Throughout the design process, a triangulated grid was assumed. This is because triangles allowed the highly articulated form of the roof to be described and can always be planar. Triangles are also self-restraining and thereby structurally desirable local geometries, particularly in such a wide spanning structure. However, triangulation could be more costly: more steel mullions were needed and there was more glass wastage when cutting triangular glass panels, as opposed to four-sided ones.

In addition to the original project brief, a grid optimisation exercise was carried out. The exercise entailed an iterative procedure, scanning through the grid elements of the model and defining four-sided planar elements instead of three-sided elements where possible. A warping factor was introduced which allowed construction tolerances to be incorporated. The warping parameter allowed the testing of different tolerance scenarios. The architects were curious to see what the layout of the grid would look like and how many triangles could be replaced by four-sided elements. A combination of three- and four-sided elements could have resulted in a cost saving. However, the rectangular elements needed thicker steel support beams, as they were quite large. To maintain a homogenous appearance, the depth of all elements would have had to be increased. Understandably, this was not in the architects' interests. For aesthetic reasons, the architects also preferred the continuous triangular pattern over the mixed version.

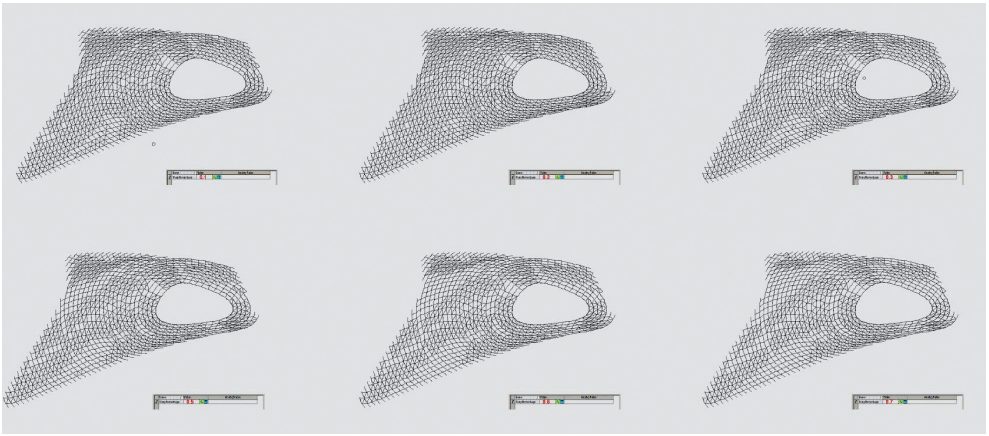


Figure 66
Grid optimisation. Using the warping factor to determine which grid cell can be a rectangular quasi-planar element and which has to remain triangular.

4.7.4. Summary

The roof of The Bergen Academy of Arts and Design is an example of where the form of the design was defined by physical forces, which resulted in the minimum necessary structural material. This was only made possible because the architects were interested in making structural performance the driving factor of their design. The project encouraged the GG group to communicate the structural performance parameters and manufacturing constraints, and thereby vividly influence the design. The boundary, and the location and definition of supports, determined the complexity of the shape, thereby satisfying the main criterion of the architects for an interesting roof articulation.

Achievements

- Generation of geometry as a pure response to physical forces.
- Communicating structural performance criteria and manufacturing constraints.

Relevance for the research project

- Setting up a design system with fixed and flexible parameters, thereby producing a complex form.
- Defining rational parameters as form-driving factors.
- Exploring the dependency between the boundary conditions and the resulting form.

4.8. Summary of case studies

From the engineering side, the projects were governed by the following four hard objectives:

- structural performance
- manufacturing

- construction sequence
- cost.

The soft objective was:

- supporting the aesthetic design intent.

Informing the building design using these objectives often helped the architect to take design decisions. The art of communicating these technical concerns determined whether the suggestions were taken on to help narrow the number of possible design options in the decision-making process or were understood negatively, as intrusion.

4.8.1. Project briefs and problem pattern

There were a number of recurring tasks involved in realising these projects. These included the definition of rational fragmentations and the creation of physical offsets from design surfaces. A further repetitive task was the post-processing of design models to allow for the transfer between modelling and analysis applications. The following list identifies the repeating tasks and the projects to which they were associated

1. To fragment a building form into buildable parts and tessellation, including the definition of node elements and connections with defined restrictions (see OVO and MT).
2. To define the geometric offset from a design surface, which takes in physical criteria such as the roof build-up and the definition of the structural offset (see MT and HCJ).
3. To post-process design models, to ensure transferability between different applications during the iterative process of the analyses and design alterations (see MT and NHI).
4. To produce design alternatives, i.e. families of solutions (see OVO, MT, ZH, NHI and NAA).
5. To form-find or form-optimise a building hull, with respect to its structural performance, in relation to particular boundary conditions and constant gravity. The aim here was to achieve an elegant form which optimally transferred forces, so that it could be built with the use of minimum material and support structure (see NHI and NAA).

The project work at Buro Happold revealed recurring problems in the design and realisation of complex and freeform envelopes.

Three of these patterns were

1. Fragmentation of complex forms and freeforms.

2. Consideration of material offset.

3. Accurate modelling, and transferability between modelling and analysis software.

4.8.2. Fragmentation of complex forms and freeforms

The post-rationalisation of tessellation on elaborate form designs was difficult, particularly when further geometric constraints were applied. This was the case with the façade of the OVO Hilton, where the tessellation had to be planar, but also had to line up with the wall partitioning and the floor levels. There was an assumption that the complicated form could be made more economical, by employing a regular and seemingly simple mesh pattern for the breakdown into building parts, but this proved to be incorrect.

The desire for planarity in the OVO Hilton project enabled the use of planar material, but it did not make the building cheaper. That was firstly because the post-processing of the initial geometry model took a long time; secondly, although manufacturing of the façade elements was planar, the complication of the geometry was shifted to its outline and connection to neighbouring parts. Therefore, each element shape was different and each angle where they met neighbouring parts was different as well; one order of geometric logic was placed on top of another. If the fragmentation of an envelope has nothing to do with the intrinsic geometric rules of the global envelope, then the result is a hybrid and is likely to be geometrically complicated.

There are clever procedures in place to place one order on top of another one. However, if this happens for the purpose of rationalisation, then the question remains of whether these considerations are actually serving this intention. The best subdivision might still be the one that immediately follows the logic of the geometry. The case studies suggest that if certain material and manufacturing constraints are known, the design can be created with these constraints in mind, using corresponding geometry sets or procedures. Examples would be the proposition to use translation surfaces for the OVO Hilton project or the regular grid development in the Zagreb Airport project. With the Museum of Transport, for example, it did not seem natural to place an abstract, regularly patterned mesh with equally sized mesh cells onto an organic freeform surface. It is of course possible, and akin to Morphosis' example on the San Francisco Federal Building, that the geometric conflict can be used as a design feature. However, it is desirable to cultivate the awareness of the dependency between the local and the global geometries, so that decisions can be made consciously. The problem pattern shifts attention to the advantage of designing with geometry sets or geometry systems, in which the parts and the rules for their assembly are known.

4.8.3. Consideration of material offset

Another recurring challenge is the physical offset of many elaborate design surfaces. They often work as long as no material thickness is applied; the edges match up and the overall appearance is right.

The difficulties appear while creating the offset, i.e. the material thickness, the roof built-up and, if necessary, the structural layer and internal skin. From an engineering point of view, it would be an advantage to work from a middle surface, and to offset outwards and inwards. However, the architects of the presented projects all preferred control of the external skin. One problem that can arise when trying to define a regular offset from an external design surface is that the edges do not meet, and consequently the structural nodes and rods are not easily defined, as in the case of the Madrid High

Court project. The offset can also have an unwanted shape, e.g. in the first offset trials for the Museum of Transport in Glasgow. The extreme changes in the angles along the ridge cause a regular offset, that is normal to the surface, to jump. This shows that it would be advantageous if the physical built-up was considered during the design conception.

4.8.4. Accurate modelling, and transferability between modelling and analysis software

A lot of time is spent on post-processing a design model to enable it to be transferred into analysis software, to apply optimisation routines or to run automated procedures for detailing.

The transfer of design models between different modelling and analysis applications was a recurring difficulty. One problem was geometry distortion that occurred when the analysis application did not support the geometry type, as was the case when a NURBS model was exported from the modelling environment Rhinoceros to Autodesk's Analysis application, Robot.

Another problem occurred when design models were converted into geometries which were suitable for rendering, but not for design development or analysis. An example of geometry which is not suitable for design development is subdivision surfaces, produced using Autodesk's Maya by converting real-time low-resolution models into high-definition models for rendering. The transfer of these models led to them becoming inaccurate; parts of the model had to be stitched together and overlaps corrected, or whole geometry sections had to be remodelled.

The process of post-processing a design model can take from a couple of hours to months, depending on the state of the design model when received, and the communication of the design geometry and modelling procedures. During the period in which this study was undertaken, Building Information Modelling (BIM) became an important topic. It promotes a data model which is shared and edited by all parties involved in the process. In the project work, however, different models of the same design were developed and data had to be transferred between numerous applications. It is desirable to allow for the flexibility of the model, and subsequently an efficient workflow, through the use of a compatible geometry representation between necessary applications, accurate modelling procedures and good communication.

4.8.5. Conclusion

The common goal in the projects was to understand and maintain the aesthetic design intent of the architects, and to keep the cost within budget. There was not necessarily a conflict in creating an intricate design, and at the same time satisfying the detailing and feasibility. New Holland Island, Zagreb Airport and the Academy of Arts in Bergen are examples of the shared exploration of geometry. The early involvement of performance and realisation mattered in the initial design formation, and resulted in fast and satisfying design development. The GG group was involved in the OVO and GMT projects to retrospectively rationalise the geometry. With these it was important to raise awareness about the potentially negative impact of the apparent, but not actual, rationalisation procedures proposed by the client.

The problems that the GG group worked on arose from inexperience with the type of geometries that were designed. All of the projects helped in understanding desirable objectives for the design implementation project documented in the following chapter.

PART 2

DESIGN IMPLEMENTATION: FORM GENERATION AND FORM OPTIMISATION - A COMPUTATIONAL DESIGN SYSTEM INFORMED BY REAL-WORLD PARAMETERS

The previous three chapters provide a picture of the state of the art in generative design system development and the engineering of non-orthogonal architectural surface design.

The investigations of reverse engineering elaborate form and the experience gained through the case studies formed the basis on which a design implementation study was created.

In the following six chapters, the development and testing of a design tool prototype is presented.

The system allows the user to generate and optimise componential surfaces.
The process integrates the buildability and the detailing of architectural surfaces, supporting the motivation to bring form generation close to real-world applications.

The constraints taken forward from the preceding study are

- Coherence between local and global geometry orders.
- Discreteness of the geometry i.e. a clear description of all geometric elements.
- Planarity of the local geometries is not an issue, their articulation can be complex, but the number of different types is controlled.
- The physical offset is taken into account.
- The interface to the neighbouring parts is solved.

Tool architecture

The tool consists of two interlinked mechanisms or engines: the growth of surfaces, and the search and optimisation of them. The first mechanism of the two engines is the geometry system. The growth engine allows the generation of a wide range of envelopes under constraints set by the user. The system employs a discrete but parametric geometry set, from which global geometries are grown. It creates a solution pool of global envelope structures. This part of the system is an inductive, bottom-up process and can be used as an independent design tool.

The second mechanism is an optimisation or search engine, which complements the growth mechanism. It explores the pool of generated envelope solutions for

suitable candidates. The surfaces are optimised using an evolutionary technique, to closely match the fitness criteria set by the user. This mechanism of selection and improvement is a top-down (deductive) approach. Both mechanisms are described in the following chapters.

The following four objectives are implemented as parameters

- Control of the number of building components (surfaces, connections, rods and node connections).
- Transferability between modelling and analysis software: flexibility of the design model.
- Consideration of physical offset from the design surface (roof build-up and structural offset).
- Representation as surface and frame structures, used literally or as placeholders.

The fitness criteria employed are:

- surface area
- number of holes
- attractor points
- undulation.

Implementation

The tool is implemented as a plug-in to Rhinoceros using the C# language. The fact that the plug-in sits within a widely used CAD application has the advantage that the user can post-process any outcome of the tool directly. The drawback of not being a standalone system is that the plug-in has to be updated constantly to operate in the new versions of the host application.

Structure of Part 2

Part 2, which describes the research project, is organised into five chapters. The first chapter is Growth Model Development. It shows the experiments carried out to derive the Unit Cell growth model, which is described in the second chapter. Surface Approximation, the third chapter, is a validation study, in which the principle of the Unit Cell concept is applied to a predefined surface. The fourth chapter depicts the implementation of the bottom-up growth mechanism to generate surfaces. The last chapter in this part is Search and Optimisation. It comprises the embedding of the bottom-up surface generating mechanism into a search and optimisation engine that employs a genetic algorithm.

CHAPTER | 5

GROWTH MODEL DEVELOPMENT

5

CHAPTER | 5

GROWTH MODEL DEVELOPMENT

5.1. Introduction – problem statement

The industrial review of this thesis documented the following recurring problems when reverse engineering elaborate structures. First, the difficulty to aesthetically, structurally and economically break global geometries into buildable parts, and define the corresponding structure (tessellation and physical offset). Second, the fact that the planarity or repetition of building elements (surfaces/connections) imposed onto the form retrospectively often resulted in forcing one order on top of another. This is true unless the form can be changed a lot and the geometry can be rationalised. Complication and thereby cost is often only shifted, not reduced.

The case studies showed that there is an advantage in using discrete geometry, so that the created form is transferable between different applications. Also, it is desirable to know that the physical offset (surface build-up such as material, structure, services, and external and internal skin) of the articulated form is coherent, and that the structure and connections can be defined. These aspects were brought forward to the design implementation, which is documented in the following five chapters.

5.2. Developing a discrete geometry system with real-world parameters

In this study, the design implementation was the development of a digital form-generating and optimising design tool, enabling the user to create architectural envelopes with a geometry system informed by real-world parameters.

The main element of the generative or growth process was the geometry system, which provided discrete components or, better, a controlled framework for the definition of distinct elements. It is called 'growth model', because it is the representation of a digital growth process. The aim was that the system was informed by real-world parameters obtained from the Review, the Case Studies and a number of independent geometric exercises. The latter are presented in this chapter.

The growth model consists of components and rules for how these parts can connect to each other. The main criterion was to design an architecturally integrated system. It meant that the growth model represented explicit building parts or placeholders, as well as how these types of components were assembled, without the necessity of an explicit material association.

The examples of geometry sets described here show that complex form catalogues can be derived from a specified and constrained geometry set. The author could have decided to use any existing form set, such as translation surfaces, which have the economic advantage that flat panels can be extracted. However, the next exercise, in which a saddle translation surface was subdivided, shows the difficulty is taken out of the components, and transferred onto the edge-joints and nodes. The author placed effort in trying to contribute to a different set of economic parameters by freeing the criteria of planar shapes, while controlling the number of element forms, shapes and

connections to the neighbouring parts. The focus of this approach was the control of repetition of both the components and their connections.

The geometric set which supported global structures made of repetitive parts demonstrated a geometric logic, i.e. the possibilities and constraints of restricted form sets. There were two reasons for why it was desirable to have control of the number of local parts.

The first reason was that repeated manufacturing routes, as well as reusable customised tools and moulds, are cost attractive. In manufacturing, it is still best to have few repeating steps in the production process and expensive moulds can be reused, unless a design can be fully cut out in 2D in a computer-aided laser or water jet. 3D rapid manufacturing, including both additive and subtractive manufacturing, does allow relatively faster production of complex parts then if they had to be produced using analogue methods. However the making of many different parts is still not feasible. The process is relatively expensive (additive and subtractive manufacturing) and there is still a limited material choice (additive manufacturing).

The second reason was that the cost and performance of the employed system, in respect to the manufacturing and assembly of the parts, can be bettered over time.

5.3. Objectives for the development of a geometry system

The objectives of the geometry system/growth model were as follows:

- The growth model should allow the creation of articulated global surface geometries, meaning that they could be doubly curved and folded.
- The elements of the growth model should represent buildable components/ parts. The considerations should be the making of the parts, the necessary material offset and the connections, i.e. the angle to the neighbouring components.
- The number of differently shaped elements and connections should be controllable (parametric), so that the number of components could be low or high, depending on the economic constraints and the design preferences.
- It should be similar to using a LEGO brick system, in which the same set of element shapes could create a wide range of different global configurations.
- The elements did not have to be planar.

5.4. Association between local and global geometry – geometric exercises

The question was what the components needed to look like to describe these global geometries and meet the listed criteria.

To answer this question, the author conducted a number of experiments. On first sight they may not appear to relate to each other much, but central to all of them was the study of the relationship between the local components and their global

configuration. The aim of the tests was to obtain the starting point for the development of a geometry system, the Unit Cell, which was then used in the tool design presented in this research.

The guideline for these experiments was the growth model used in the ECSS project. The ECSS geometry set consists of only six fixed shapes, all of which were developed from a planar square. Each element or tile was defined within the same-sized three-dimensional box frame. The implementation of the ECSS growth model was hard-coded, meaning that the elements and their rotations were predefined. The user of the tool only has the choice to specify which of the six elements to employ in the generative process. The question was whether there was an alternative to creating a geometry set within a constraint frame. Did the elements need to be defined within the same matrix or unit frame, in order that they could be recombined, and produce closed and variable surfaces from the same parts?

Therefore, the experiments either maintained the relation to a grid or a frame; otherwise the dependency to a grander frame network was consciously avoided.

The experiments exhibited three different approaches. The first approach was a top-down one, breaking the global geometry into local parts (experiment 1). The second was a bottom-up one, predefining the local components and the connection rules (experiment 2). The third was a bottom-up one as well, but this time defining components within a reference matrix (experiments 3, 4, and 5).

Subdividing a saddle surface

In this experiment, a regular saddle surface was tessellated for observing the resulting parts and the relation between the parts.

Growing form the naïve way

Surfaces were generated by connecting predefined triangle shapes using a restricted range of connection angles.

Grow triangles

Triangulated surfaces were grown by placing one vertex at a time, allowing the z-position of the vertex to vary only in a specified range of unit steps in relation to the positions of the neighbouring vertices.

Grow quads

A small number of tile shapes, which were defined in a grid cell, were used for growing surfaces in a linear growth order.

Grow surfaces within a sphere network

Triangulated surfaces were grown within a three-dimensional lattice of a closely packed sphere assembly.

5.5. Experiment 1 Subdividing a saddle surface

A saddle surface is a controlled, doubly curved surface. The term 'saddle surface' is derived from the shape of a horse saddle, which curves both upwards and downwards. The so-called 'standard saddle' is a hyperbolic paraboloid, a smooth surface in the Euclidean space $z = x^2 - y^2$ of the second order. Saddle surfaces have a negative Gaussian curvature. A saddle surface can be created from a minimum of four planar elements (Figure 67).

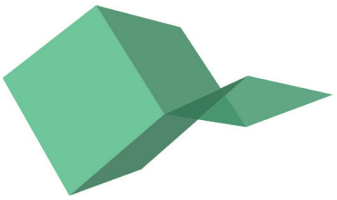


Figure 67
Minimal saddle. Describing a saddle surface with four planar components.

A saddle is a translation surface, where for example two arcs are translated one (generatrix) along the other (directrix), so that the resulting mesh is created by spatial, parallel arcs in two directions (Figure 68). The apparent subdivision or tessellation, therefore, is to use the construction curves of the translation. The advantage is that every translation surface can be subdivided into planar parts.

From the Review chapter in this thesis, it became apparent that planarity was important where glass was the preferred building material and high transparency in the structure was desired.

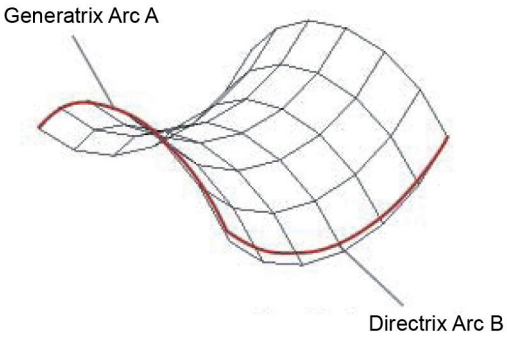


Figure 68
Saddle translation. A saddle surface created by translating two identical arcs.

The task was to analyse the relationship between the local parameters of the mesh components, understanding each mesh cell as an independent part or component. The lengths of the edges were documented, as well as the angle at every corner and the angle to the neighbouring components. As the saddle had two symmetry planes, it was sufficient to examine only one quarter of the whole surface (Figure 69). Note that this was only true for examining the shape of the elements. In order to determine the number of necessary parts to construct the saddle, the number of elements had to be doubled. This was because the reflection symmetry did not mean that a particular element became the element on the other side of the symmetry plane by a simple rotational transformation. It had to be mirrored and in manufacturing terms this

means that it is a different element. The experiment was conducted using Autodesk's computer-aided design (CAD) package AutoCAD.

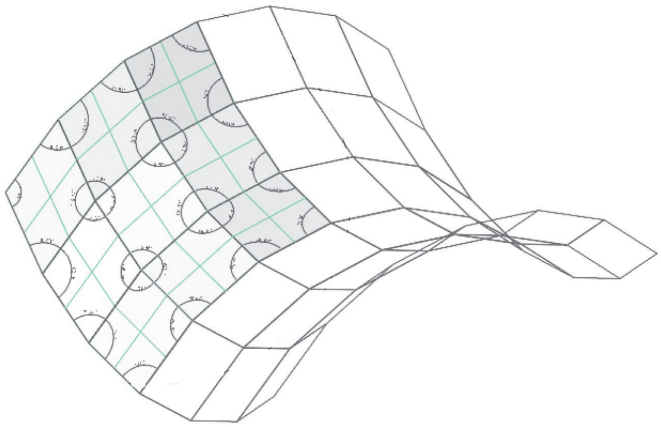


Figure 69
Symmetry of the saddle surface. It implied that only one quarter of the whole surface needed to be examined.

When measuring the angles between two edges, thus the corners in the individual component, the diagonally opposite angles were always the same. Some components shared the same pairs of corner angles, therefore had the same shape and surface area, and thereby were identical (Figure 70). The angles were measured by placing a coordinate system using three points, taking into account three vertices of the rectangle which defined its plane.

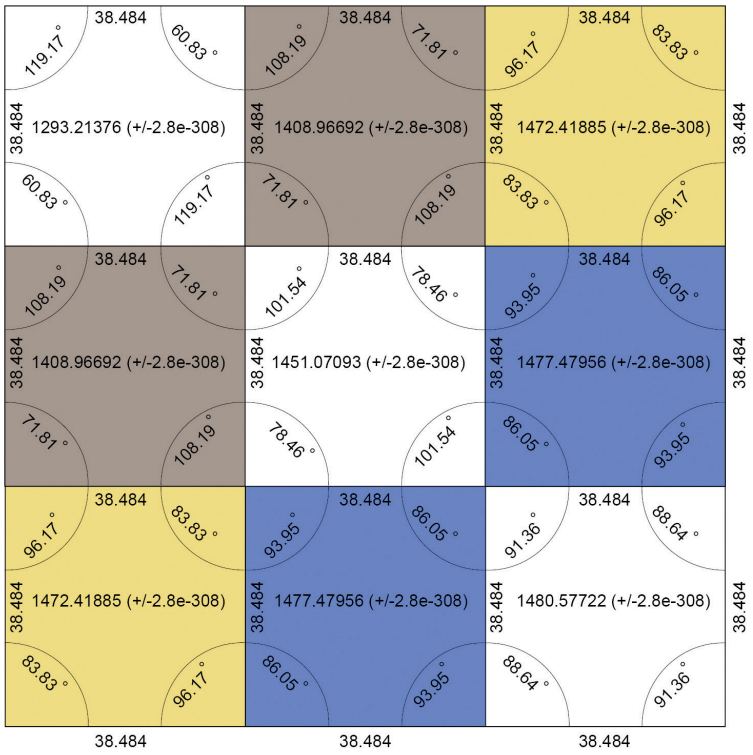


Figure 70
Saddle plan. Parts with the same colour are identical. White fields indicate components which are not repeated.

Then the angles between neighbouring components were measured. From the middle of an edge between two faces, lines were drawn to the opposite edges. A coordinate system was placed at this edge, by creating a plane through the endpoints of both lines and the middle point. Then the angle was measured. All angles were different (Figure 71).

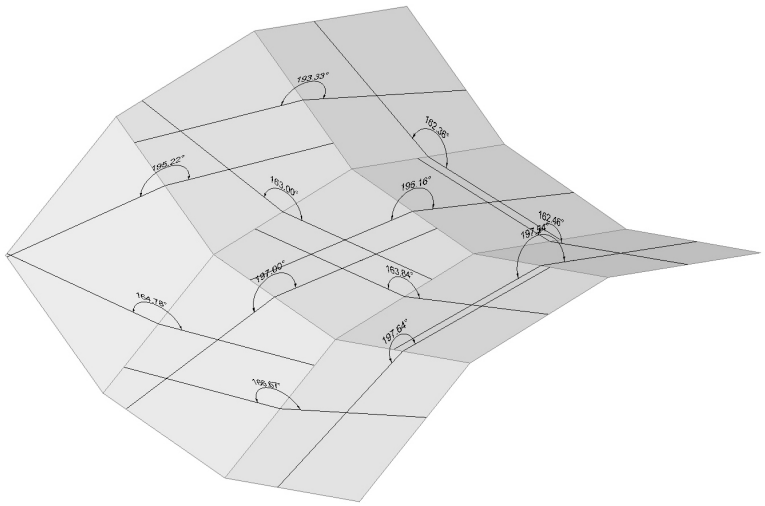


Figure 71
Angles between parts. The angles at which the parts connected with each other were all different.

Summary

All elements which formed the saddle were planar and all edges of the components had the same length, but the shapes of the elements were only partially repeated. Here, the constructed saddle had a total of thirty-six components. It could be constructed using twelve repeating shapes. Nine elements and six shape types were needed to describe one quarter. The shape types of this one quarter had to be mirrored in order to describe the whole saddle, so six more element types were added.

All angles between the components in the quarter that was examined were different. Therefore, it could be said that the symmetry and the partial repetition made the surface relatively economic, in terms of manufacturing. However, this was only true regarding the manufacturing of the components, not so much in terms of the connections. The elements could not be reused to form different geometries, other than a reflection of the symmetry parts (Figure 72). When considering structures other than steel mesh and glass, such as space frames, and concrete or composite structures, we need to consider that a repetition of the angle in which the components meet has an effect on the cost of a structure, because of the manufacture and construction of connections.

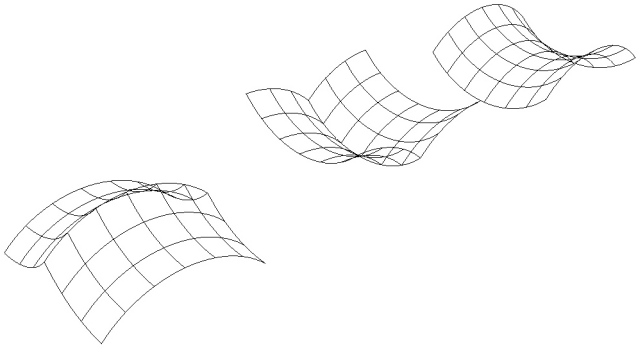


Figure 72
Variations of the saddle.

5.6. Experiment 2 Growing form the naïve way

5.6.1. Automated growth of triangles

As the control of the number of elements and connections was the driving criterion, the obvious test was to define a single element shape and decide on a range of angles in which the same-shaped parts could connect. This is probably how everyone starts when setting out to 'grow' surfaces.

In this exercise, scalene triangles were connected in a linear growth order by describing the connection side and the angle between the two respective planes. Each triangle was the same as the first, just rotated and added at an inclined angle. The range of angles at which the triangles could connect to the neighbouring ones was decided to be 0°, 15°, or -15°.

The main concern of the experiment was to free the growth from an underlying grid, hoping for more interesting surface developments. Here, polar coordinate systems were employed for every triangle element.

The process was implemented using GC's scripting language, GC script.

Implementation

The initial triangle was drawn as a shape, and had vertices labelled pole 0, pole 1 and pole 2, based on the vertices (points) in the order of creation. Hence, whenever this triangle was redrawn as a new tile, the points had to be given in the same order, so that we knew which ones were pole 0, pole 1 and pole 2. The sides were labelled according to the poles: side 0 between poles 0 and 1; side 1 between poles 1 and 2; side 2 between poles 2 and 0 (Figure 73). The function needed the following inputs: a side to add the next triangle to and an angle to join the two triangular planes to each other. These were given as two lists.

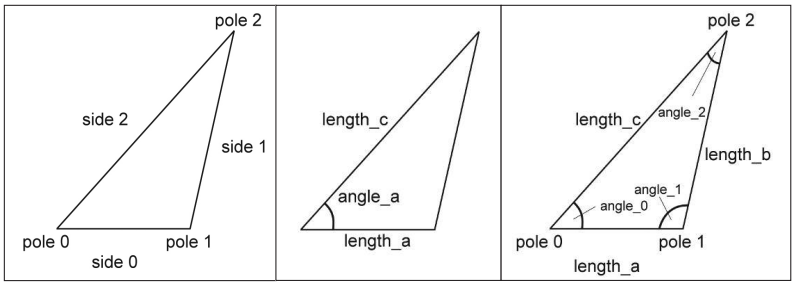


Figure 73
Single triangle shape for the growth process.

The triangle was drawn with only the given information in the diagram in Figure 74, because the other information shown in the right-hand side diagram in Figure 73 was determined using existing functions in GC, labelled 'length' and 'angle'. Having found all the information about the triangle, all three points were plotted, taking any one of the three vertices as the origin. Polar coordinates were employed, as the distance and angle were known.

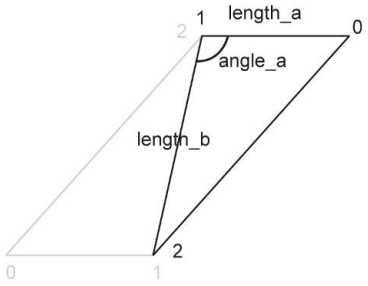


Figure 74
Placing the next triangle. It was connected along the side between poles 1 and 2.

Outcome

Figure 75 shows that while the first row of triangles created a continuous band of triangular shapes, by the beginning of just the second row, the new band of triangles did not connect to those of the first; the structure branched. The diversion was particularly obvious when a linear growth order was used.

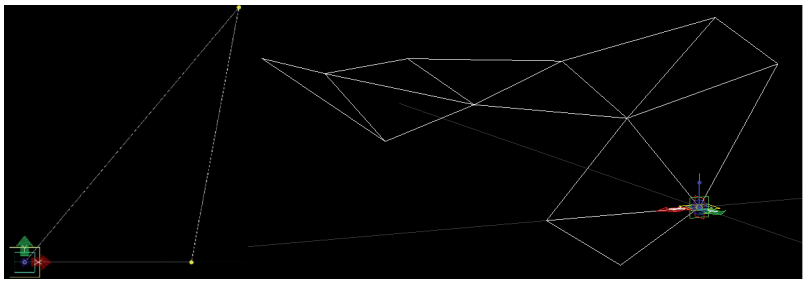


Figure 75
Growing triangles. The first triangle (left), and eight more added to different sides and at different angles of 0°, 15° or -15°.

Summary

The above process did not allow the creation of closed surface entities and the structures branched out. To achieve closed surfaces, individual elements would need to be defined to fill the gaps between the branches.

5.6.2. Manual experiments: circular growth of triangles and linear growth of squares

To avoid a time-consuming implementation process, two more exercises were conducted manually. The CAD package Rhinoceros was used for drawing the model. The previous automated test showed that the structures did not reconnect when the second row started growing. A circular growth was therefore tried out. This time, the shape of the element was an equilateral triangle. Triangles were manually placed along a circular path. Akin to the previous experiment, coordinate systems were placed locally to measure the angle at which the next triangle could be connected. The circular growth order reduced the distance between the subsequent triangles, but the field could still not be closed (top image of Figure 76). In a further trial, the base component used was a square. Every placed square was the same as the other. The angle at which the components connected was within the same range as in the previous experiments (i.e. 0°, 15° or -15°). Once again, the outcome did not produce closed surfaces, but led to branching and overlap. The branching can be seen in the bottom image of Figure 76.

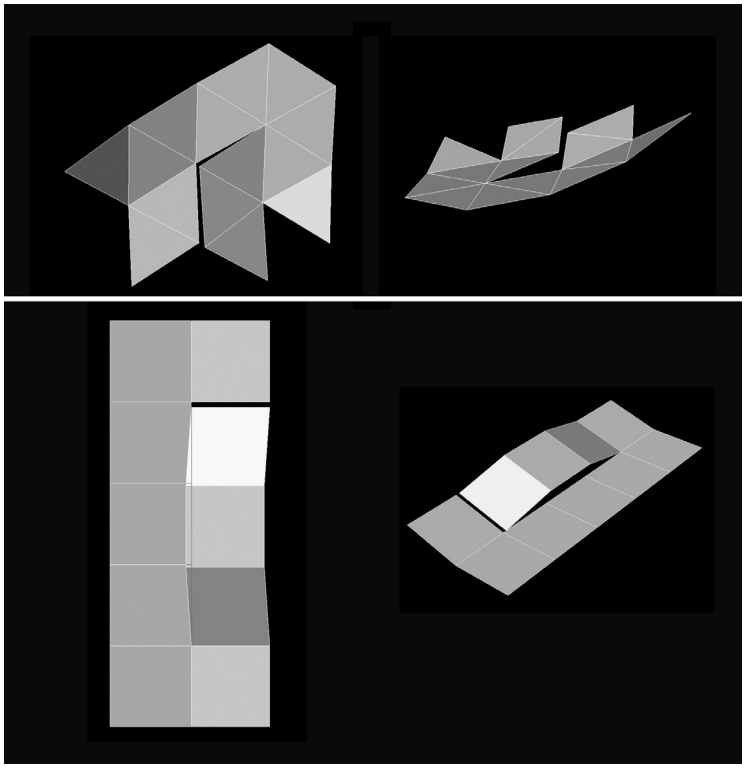


Figure 76
Growing form the naïve way. Experiment results: branching, and in the next step, overlapping structures.

Summary

The experiments showed that the circular growth order did not change the fact that with a fixed element shape and angle of connection, no continuous surface fields could be grown. The tests suggested that there was a need to define the relationship between the shape of the element, its positioning within its particular coordinate system and the corresponding catalogue of possible global geometries. Therefore, the question remained what this relationship between the local and the global geometries could be. What did the individual component need to look like and how was it connected to allow for different continuous entities? A naïve approach did not prove useful.

5.7. Experiment 3 Grow triangles in a grid

Description

In the following experiment, the growth happened by placing points in a linear growth order across an equally spaced X-Y grid, with the following rule: no two points on any triangle may differ by more than the 'range' units in the z direction.

The range was a variable in the implementation. The process employed a global coordinate system.

The main concern in the development of this process was to define an appropriate growth order, meaning the sequence in which the points were placed, that would allow the creation of a closed surface of triangles.

In plan view, the grown triangles formed a regular grid of right triangles, as shown in Figure 77. The triangles were drawn when three points of the respective triangle were defined.

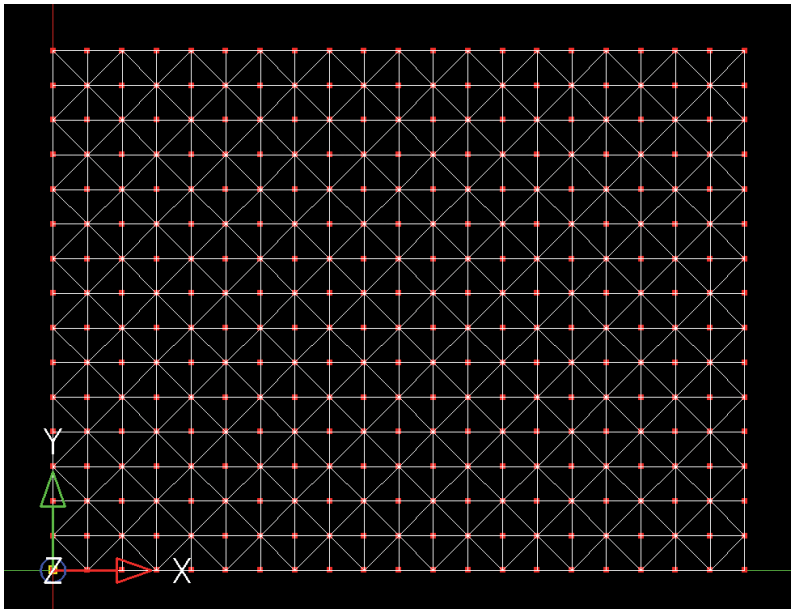


Figure 77
Field of triangles grown in a grid.

How it worked

The grid was grown linearly by placing one new point at a time, which created one and, in some positions, two new triangles.

It began by growing along the bottom row, from the bottom left, and new points were created in the order shown in Figure 78.

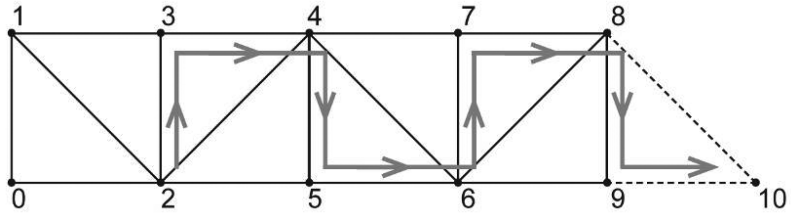


Figure 78
Order of placing points.

When creating each point, the rule that no two points on any triangle should differ by more than the 'range' units in the z direction had to be satisfied. Range 1 was employed (Figure 79), meaning the level change could only be one up or down. The two existing points, which were one side of the new triangle, needed to have their z values read, in this case points 8 and 9. They did not differ by more than the 'range' units in the rule. To satisfy the rule, the new point (point 10) needed to be located between itself and each of the other two points (points 8 and 9). This was possible, because being at the same height as either of the two points (points 8 and 9) satisfied the rule. However, the resulting structure would be visually unexciting. The question, therefore, was what range of possible values under the rule were available. The answer was that the new point could be as much as 'range' units

higher than or lower than each point (points 8 or 9), but it had to work for both. So the lowest that it could be was 'range' units lower than the higher of the two points and 'range' units higher than the lower of the two points. This is a positive range, i.e. the lowest possible point was lower than the highest possible one, because it was understood (as described above) that the heights of the two points, points 8 and 9, always fell within this 'range'. This is demonstrated in Figure 79.

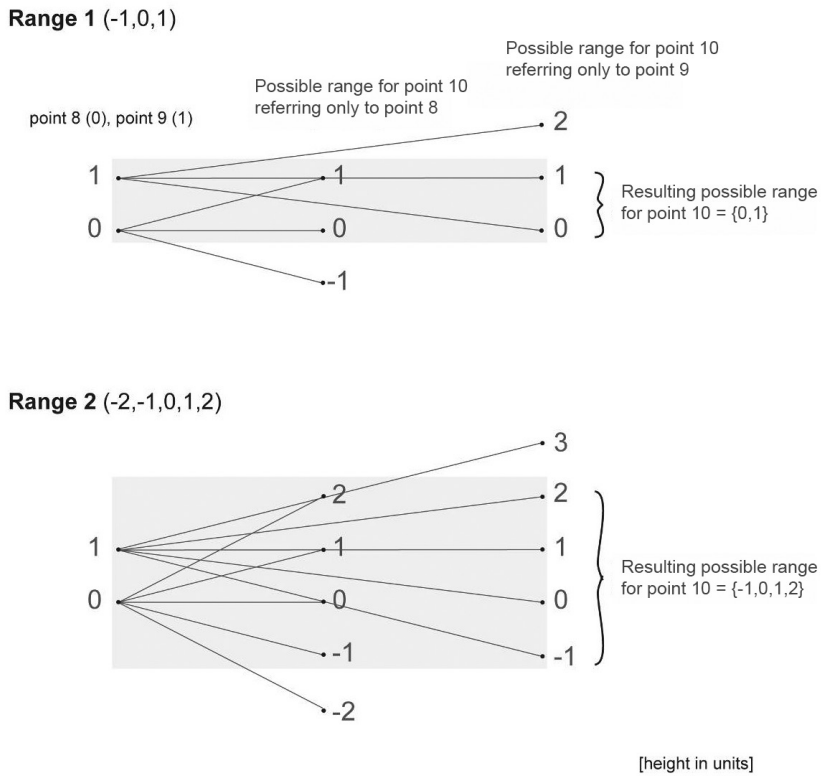


Figure 79
Finding the range of possible z-positions for the next point.

When moving onto the second row and placing points regularly from the left to the right, it was possible to end up with the situation shown in Figure 80, wherein placing one point creates three new triangles. Note that the first point in the next row did not create a triangle in this triangulation pattern. Only when the second point was placed in the new row were the two triangles defined at the same time.

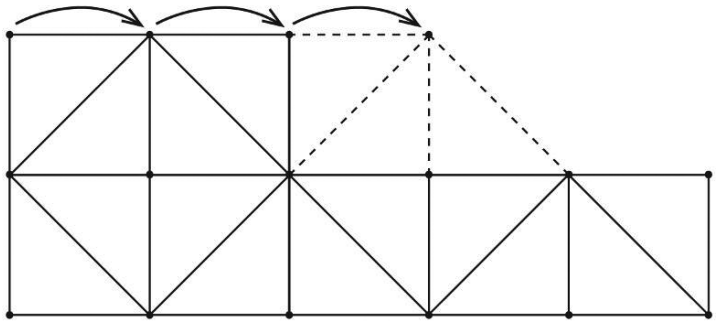


Figure 80
Placing one point creates three new triangles.

Another problem was that the height of the four points (in units) could be as shown in Figure 81 without breaking the rule, but there was no one height for the new

point which would have satisfied the rule (Range 1) with all of the other four points simultaneously, and this was not desired.

However, when building the row shown in Figure 82, followed by the steps shown in Figure 83, then each time a new point was placed, only two new triangles were created, except at the ends. In both steps, the points that were needed to satisfy the rule were connected in a string of three.

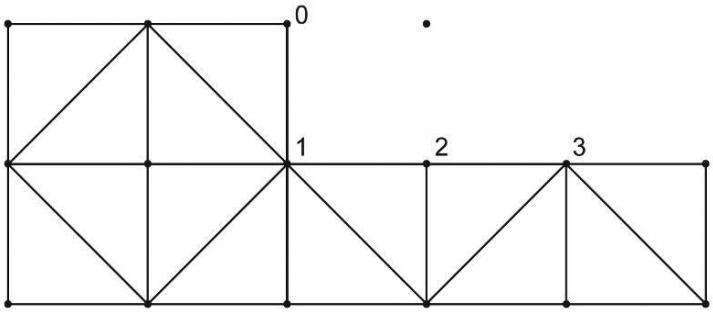


Figure 81
Finding a possible height for the 4th point.

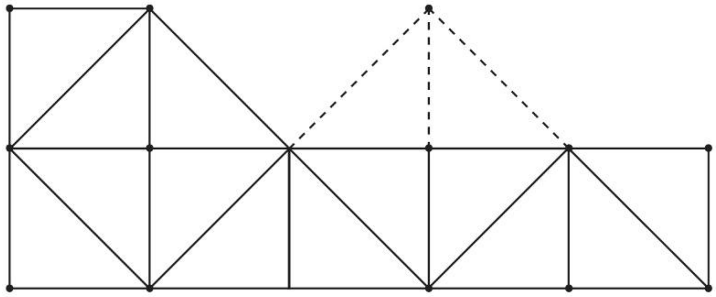


Figure 82
Alternative routine. An alternative way to build the row, by first skipping one point.

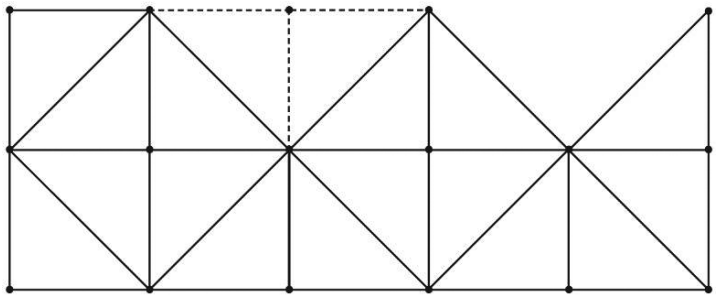


Figure 83
Next step in the alternative routine. The missing points were placed after every second point was defined.

The question was whether there was a height for the new point that satisfied the rule for all three points. The answer was yes. The height of the centre point of the three could always be used. It was understood that if the rule was satisfied between each outer point and the centre point, then it would be satisfied between each outer point and the new point. By matching heights with the centre point of the three, the rule to that point was satisfied as well.

So, what range of possible values was available for the height of this new point? Similarly to before, the new point could be 'range' units higher or lower than each point individually, and hence, the lowest that it could be was 'range' units lower than

the highest point, and the highest it could be was ‘range’ units higher than the lowest point. Once again, this range was positive, because there was already at least one satisfactory height known.

One growth step could be built on the next, using the sequence described. The end points of each row were always restricted by just two points, so followed the same method of construction that was applied to the first row.

A random selection of the height of each new point between the possible range of values, would satisfy the rule in each situation in which surfaces were generated, as in the example shown in Figure 84. Here, the range employed for the rule was Range 2.

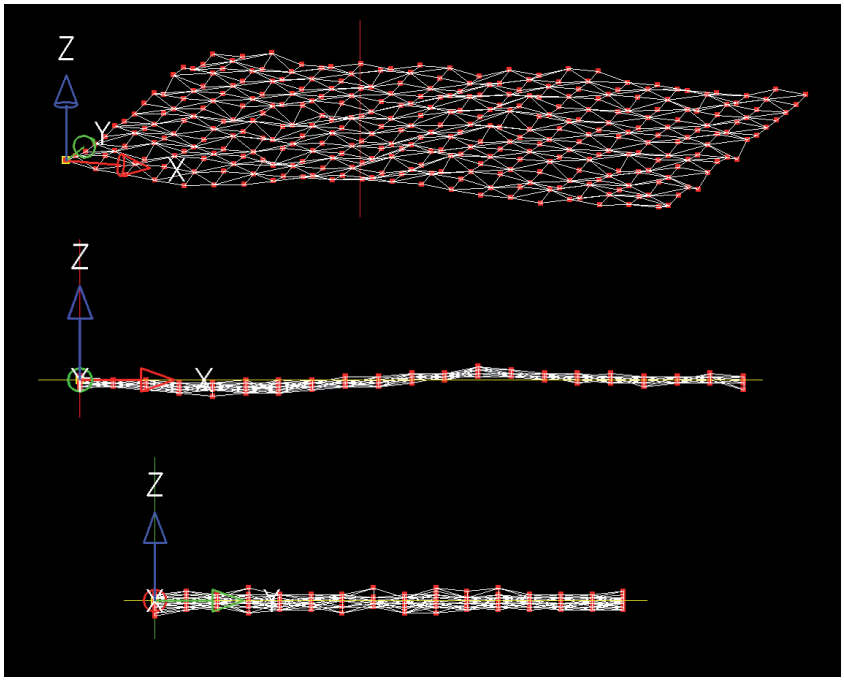


Figure 84
Surface using Range 2. Isometric view (top), front view (middle) and front view (bottom).

Conclusion

The triangles were defined by placing points under a rule, but the question remained how many different triangle shapes were used.

The pictures demonstrate the limitations of generating a surface with a linear growth order. It can be seen that from the front there is a distinctive wave-like pattern created as the surface rises and falls across the rows, but from the side, there is no evidence of this wave-like pattern. This shows that the freedom to change heights was primarily along the rows, not up them. Through analysing the construction methods, the author concluded that this was due to the freedom of the first row. Each new point created along the first row had only two points to satisfy the rule with, whereas in the next rows, new points were always trying to match the rule with three points. Therefore, the wave-like pattern generated by the first row was carried up the future rows. Alternatives could be created by looking at different growth orders, e.g. radial, thus spiralling around the starting cell. Another limitation to the above growth method was that the surface could never fold back on itself, because of the regularity in the spacing of the x and y coordinates.

5.8. Experiment 4 Grow quads

Description

In this experiment, surfaces were created using quadrilateral tile shapes and their rotations. As in Experiment 2, the building parts were defined at the beginning of the growth. The shape of the seed component in Experiment 2 was a single arbitrary triangle which did not relate to any greater network, but in this experiment the elements were defined in a grid cell: an x-, y- and z-dimensional box. Therefore, the quadrilaterals all looked the same in plan, forming a square patterned grid (Figure 85). The experiment was subdivided into two steps. In the first step, only one element shape and its rotation were used. In the second step, a new shape was introduced which closed the gap that otherwise could not be filled with the original tile shape. It was a progressive development. In the second stage, however, gaps were allowed and the growth would step into the next position. The goal was to find out how few element shapes were needed to create a closed surface. The experiment encompassed the development of the ECSS components, which were also defined in a Unit Cell. The most important part of this exercise was to develop a sort of tile factory, i.e. the automated generation of an element list and a second list which established all scenarios of possible neighbours when the program was executed. The exercise was implemented using a GC script. As the vertex displacements in the z direction were controlled by a unit step parameter, the grown surfaces could be referenced to a z-dimension of the reference grid with a unit size stepping. A unit was defined as the parameter "HeightZ", whereby the x and y units were fixed to the unit size 1. The grid size and the "HeightZ" unit size were defined as variables. The first study used a quadrilateral with one vertex elevated by one unit and its rotations, as shown in Figure 86.

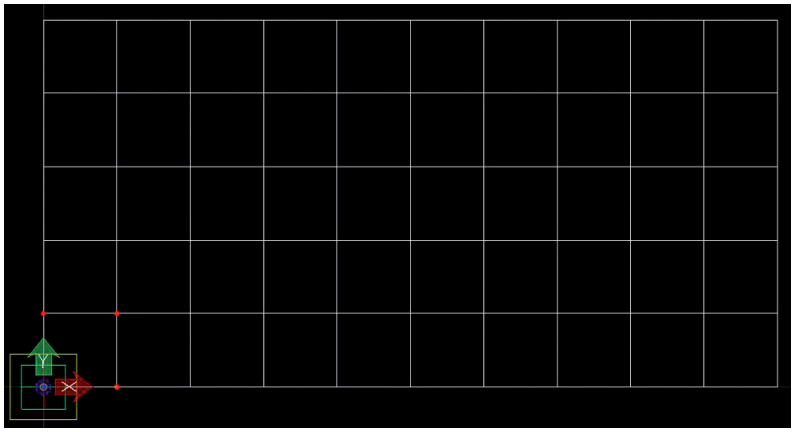


Figure 85
Plan view of grown surface. The surface was built from a small number of quadrilateral shapes.

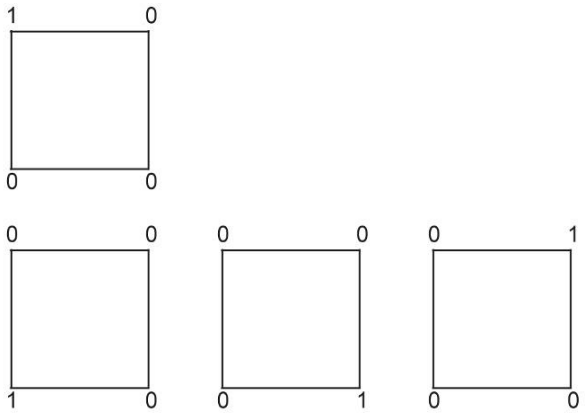


Figure 86
One tile form and its rotation. Tile with one vertex raised (top) and its rotations (bottom row).

New tiles were placed next to the previously placed ones which 'fitted together'. All the tiles were placed on the x-y plane, i.e. the vertices with three zeros on each of the tiles had the height z = 0, and the vertex with z = 1 had the height of one unit.

If tile 4 of the four above (1 in the top-right corner) was the first element in the second row, then the next tile had to be tile 1 (1 in the top-left corner), as it was the only one out of the four that fitted. If these tiles were grown linearly (going along the first row, then along the second row), matching the new tile with the left and bottom vertices of the tiles in place already, then a situation was reached where no new tile fitted in the next position to be filled. In this situation, a new tile had to be introduced (Figure 88).

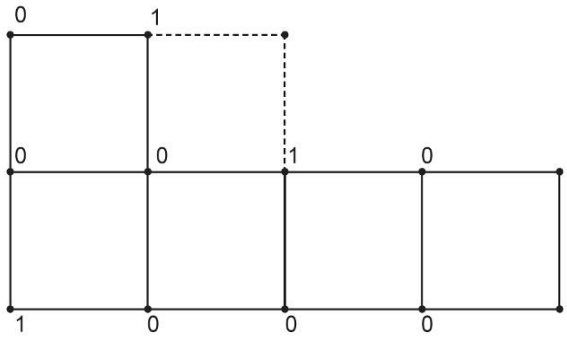


Figure 87
Empty position. A situation can be reached where no tile fits the next position.

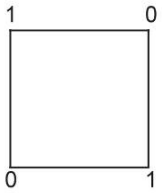


Figure 88
A new tile. A new shape was added, with two vertices diagonally opposite each other, raised by one unit.

Therefore, there was a list of five possible tiles, and with these, a grid could be filled in the linear growth order shown in Figure 89. Note that no tile was needed to grow the closed field which had three out of the four vertices raised. If there was more than one possible tile for a situation, then one was chosen randomly.

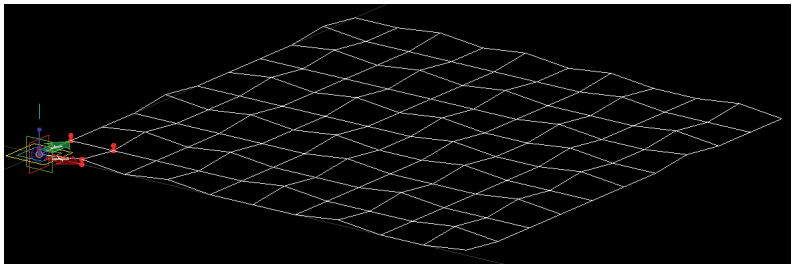


Figure 89
Closed field of tiles created in a linear growth order.

Implementation

First, a list was made with all the possible tiles. The list was used as a reference to draw a particular shape. However, if the points for each shape were stored, then a list of points could be given to create a shape (it is difficult to draw a shape any other way).

The new tile's vertices, named poles, were ordered when the points were given; the vertex at point 0 in the given list would be Pole [0], the vertex at point 1 in the given list would be Pole [1] and so on. To know which pole was where geometrically (top left, top right, etc.), the shape was always drawn in the same way, i.e. drawn anticlockwise from the bottom left. Therefore, the points of each tile had to be stored in this order.

There were five tiles stored in the list. Each tile shape was defined by its four vertices. The shape was drawn by connecting its respective group of four points from the list.

Points could have been added manually to this list for each of the five tiles, but if that had been done for all four rotations of twenty or so different tiles, it would have been tedious. Therefore, starting with only the four points of the initial shape, there needed to be a method to automatically populate the list with points for the rotations. These could be found when the points of the initial shape were rotated anticlockwise by 90°. When translated so that the bottom-left corner was at the origin (adding 1 to the x coordinates of all the four points), the vertices of the rotated tile were obtained.

However, as a list, they were in the wrong order. Pole [0] moved from the bottom left to the bottom right, Pole [1] moved from the bottom right to the top right, etc. This meant that when the points were stored in the list of possible shapes, they had to be stored in the correct order.

Figure 90 shows the first rotation, where the points had to be stored in the order 3, 0, 1, 2 (travelling anticlockwise from the bottom left). For the second rotation, the order was 2, 3, 0, 1, and for the final rotation, the order was 1, 2, 3, 0. The points were added to the list of possible shapes, as well as the fifth tile, whose rotations were not included.

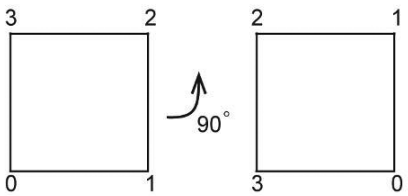


Figure 90
Rotation of the tile. It caused the point to be stored in the order 0,1,2,3, which is different from 3,0,1,2.

As a full list of the possible tiles had now been found, the attention shifted to finding the neighbouring tiles which fitted. When connecting a new tile to the right of a previous tile, the heights of the joining vertices had to be read, and it had to be checked whether they matched or not. This was done for all the possible new tiles, in order to find a list of ones that matched. This list was created once for all the possible tiles and then the tiles were recalled from it. For the five different tiles (tiles 0 to 4), the heights of the left vertices, first the bottom left and then the top left, were stored (Figure 86 and Figure 88).

There were three different scenarios (0, 1 and 2). In each one, there was one element and the list of tile numbers that matched it. For instance, tiles 1 and 4 both had heights 0 and 1 at the bottom-left and top-left vertices, respectively.

This could be repeated for the bottom edges of the new tiles to get another list of combinations, e.g. tiles 3 and 4 both had heights 0 and 1 at the bottom-left and bottom-right vertices, respectively.

The creation of these lists occupied a reasonable part of the script, but the essential steps were:

1. For each new tile, find the corresponding scenario (check heights of vertices).
2. Check the list to see whether this scenario already exists (equate the first and the second values in the scenarios).
3. If it does exist, then add the tile number to the right-hand list for that scenario.
4. If it does not exist, then create a new scenario, so that this tile can be added.

When a new tile was placed, the height of the vertices of the already placed neighbouring tile to which the new tile would be added, had to be read. This enabled the identification of the scenario, i.e. which element was placed already and the possible tiles that could be placed next to it.

The shapes were indexed using the x and y grid coordinates of the bottom-left corner. An empty grid list was created according to the size of the specified grid, with an empty list for every (i, j) location. The shapes were inserted into the correct place within the grid, as they were built. Then the pole height information was extracted from the shapes.

To place a matching tile to the right of one that had already been placed, the heights of the right-hand vertices of the current tile were read. Then the scenario was looked up that gave the list of possible tiles that could be placed at this location. A random tile was chosen from this list and drawn in the appropriate position in the grid.

When a tile had to be matched to the right and top of already placed tiles, then the method had to be repeated twice, once for each neighbouring element (left and bottom). This way a tile could be found that would fit to both existing pairs of vertices, creating the intersection of both scenarios.

Code alteration

There were a few essential changes to the setup, but they only required small alterations to the GC script. The rule that matched one tile to another was redefined. This time, a change in level was allowed, for example a tile that would connect to a neighbour with the vertices {1.000, 1.000} was allowed not only to match with {1.000, 1.000}, but also with {0.000, 0.000}. In the latter case, it would be placed such that the base height altered.

When the scenarios were equated, the differences in height between the two vertices of a side were looked at. If the positive or negative difference between the two vertices was strictly enforced, then the slopes had to be considered as well. New tiles had to be added which allowed the changes. The new tiles consisted of a new shape and its rotations (Figure 91).

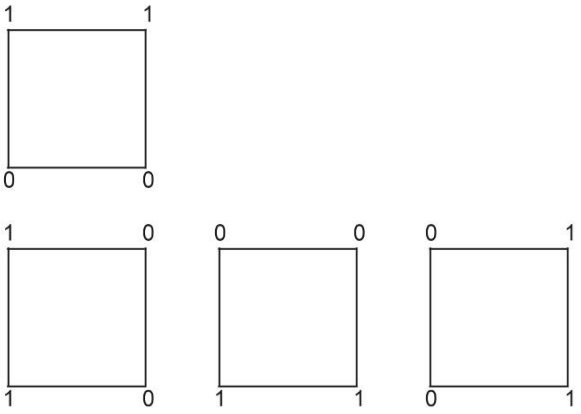


Figure 91
The new shape and its rotations.

The rest of the process was the same. If no new tile could be found that fitted in the desired place, then the position was left blank and the process moved on to the next tile.

Conclusion

The elements were defined from the outset of the growth process as in Experiment 2 and were not created on the go as in Experiment 3, yet the shape of the elements referred to a grid as in Experiment 3. In fact, in Experiment 3, the finally created triangles had repeating shapes. In both cases, the elements were distinct from each other, with respect to their vertex displacement in the z direction.

'Grow quads' became the starting point for this research project's growth model. The principle to generate a table of tile shapes and then a sub-table of the possible tiles for the respective position to be filled was taken forward. The fact that the generation of tiles referred to a three-dimensional grid cell, with the grid spacing parameters x, y and z, allowed for a parametric tile system.

5.9. Experiment 5 Surfaces within a sphere network

Closed sphere packing creates a controlled centre point network. Random surfaces which are generated within this network lead to repeating shapes and recurring connection angles. This experiment was the crucial test which led to the definition of

the Unit Cell principle, which signifies that geometry sets can be defined within any three-dimensional lattice. The Unit Cell principle will be explained in the chapter of the same name in this thesis.

A closely packed sphere configuration is a controlled neighbourhood of repeating parts and repeating relationships between the parts. When the centre points of the spheres are connected in closely packed configuration, a regular lattice is created. Within this network, triangulated faces can be defined, which in turn allows the growth of continuous triangular configurations through random selection. The purpose of this experiment was to study the resulting configurations of repeating shapes and intersection angles.

Defining the closest packing of spheres is a study in its own right and can be found in different areas of research. Examples include crystals in biology or medicine, the packing of goods in the densest manner possible to save packaging costs and materials in industry, or the not-so-contemporary problem of how to pack cannon balls in the most efficient way. In this study, the interest focused on the repetition which was expected to occur between the neighbouring elements of such a regular structure. The two most common configurations of dense sphere packing are those organised as AB_AB_AB (hexagonal close packing) and ABC_ABC_ABC (cubic close packing). Both formations achieve a 74% usage of space. In the latter, three levels of planes are stacked, translating each time, before the fourth plane is positioned directly over the first plane. The simpler configuration, AB_AB_AB, exhibits only two plane shifts before the sequence is repeated and is shown in Figure 92. To simplify matters, the AB_AB_AB formation was used in this experiment. Figure 93 shows how the position of each sphere was determined. Figure 94 presents a rendering taken when the implemented process was running.

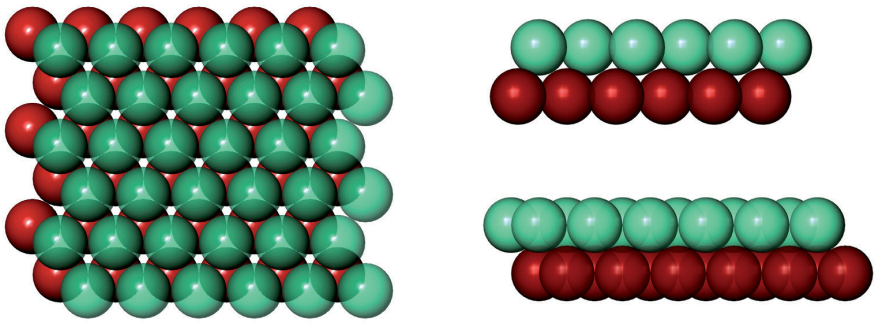


Figure 92
Hexagonal close packing. Plan view (left), left view (top right) and front view (bottom right).

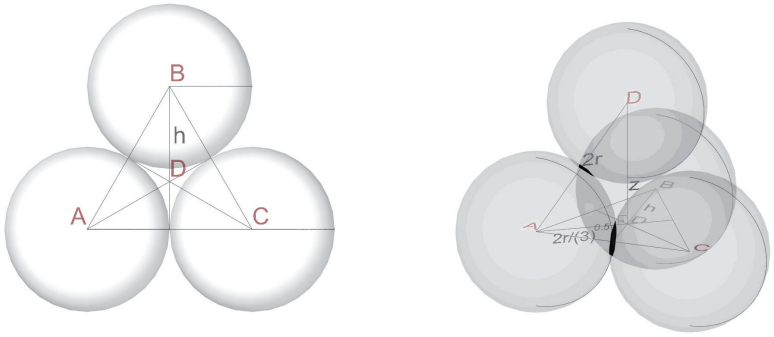


Figure 93
Packing.

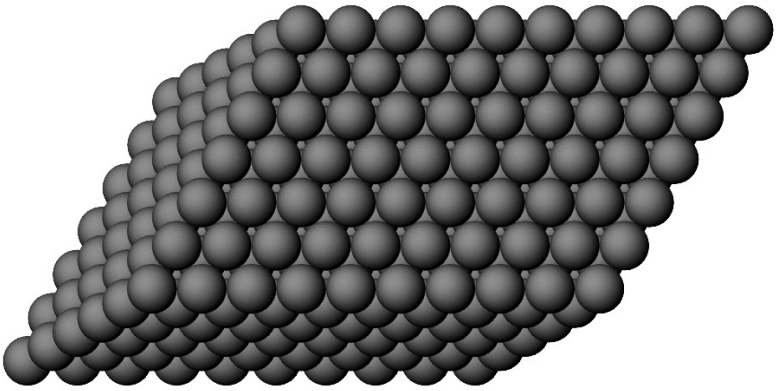


Figure 94
Block of closely packed spheres. Rendering taken from the implemented process as it ran, showing the hexagonal packing configuration.

There were many different ways to join the lattice between the centre points of the spheres across two levels, so it was necessary to decide upon one method (Figure 95). When looking at one centre point and defining the triangles which arose from connecting it with the neighbouring triangles, four planes were created. Two of these planes were oriented orthogonally and two diagonally (Figure 96).

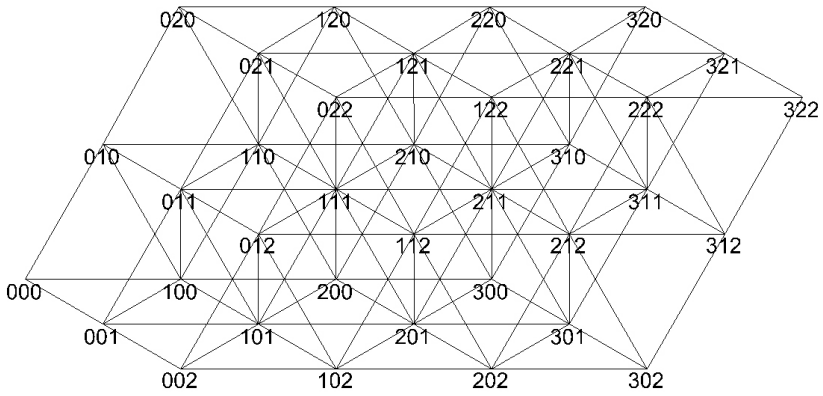


Figure 95
One sphere centre point and its four planes of triangles.

The experiment was implemented using a Visual Basic script in AutoCAD (Figure 97). The variable parameters were the size of the spheres, the number of spheres in a block and the number of triangles which were grown. The growth order, and thus the directional sequence in which the triangles were added, was linear. If more than one triangle could have been chosen to connect to the current one, the choice was random. In this way, different surfaces were obtained when the script was executed multiple times with the same settings. Two types of triangle shapes were created. These shapes depended on whether the element lay in a tilted plane or in an orthogonal one. Figure 96 shows the triangles in the orthogonal planes in red and those in the diagonal planes in purple.

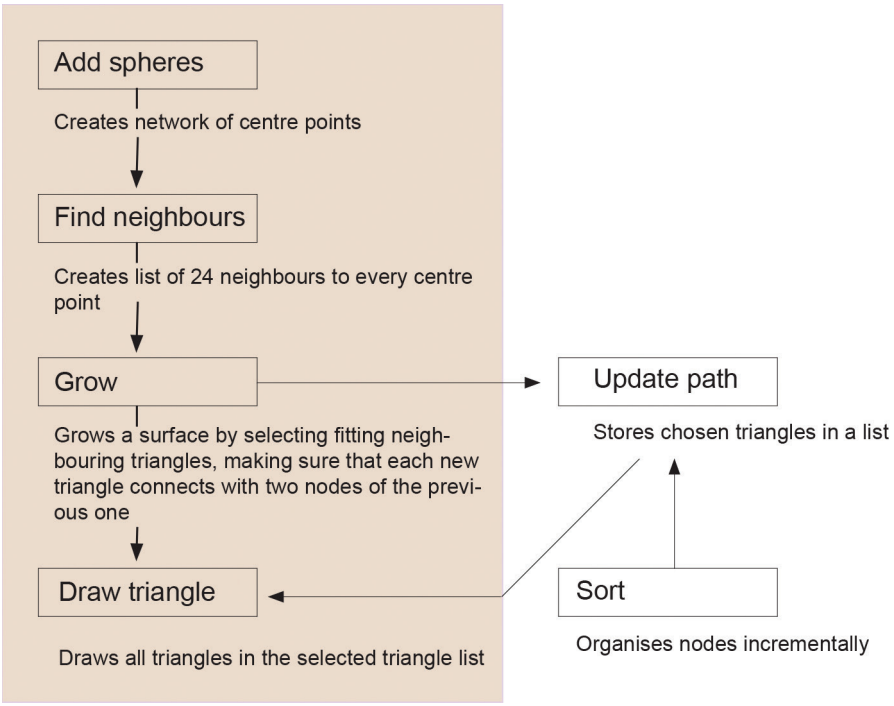


Figure 97
Flowchart: surfaces within a sphere network.

There was no checking routine which would ensure that over the next row, the surface would be closed. The structures were therefore more like continuous strings of triangles, rather than surfaces. This could have been overcome by either changing the growth order, e.g. to a circular one, or including a rule which checked the neighbourhood for compatibility. In general, however, the experiment showed that from only two element shapes, many different structures could be grown (Figure 98 and Figure 99).

The build-up of the experiment could have been repeated using three-dimensional lattices, different from the translated triangular lattice that was defined on the basis of the AB_AB_AB sphere configuration. A generalisation from this experiment led to the Unit Cell principle described in the next chapter.

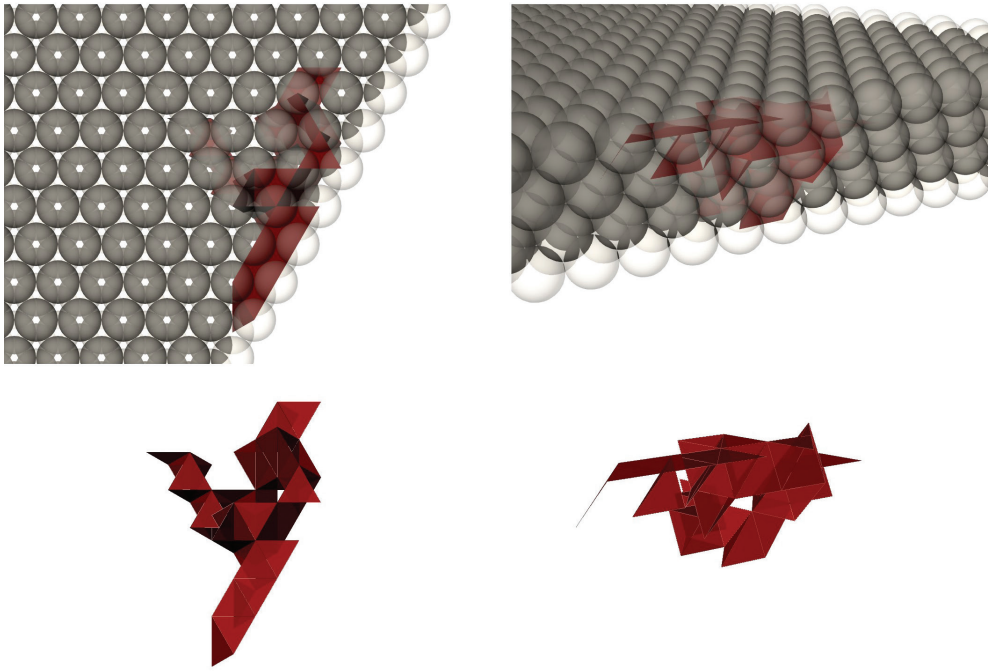


Figure 98
Sample of a surface. It was grown by random growth selection in a linear growth order. The upper row shows the structure within the sphere cluster; plan view on the left, isometric view on the right. The bottom row shows the surface in plan (left) and in isometric view (right), extracted from the spheres.

Figure 99
Sample structures.

5.10. Conclusion on geometry experiments

Experiments which have not been documented in detail here, but are worthwhile mentioning, include a test similar to Experiment 2 which used squares instead of triangles. The outcome was the same, in that the resulting structures branched out and did not create closed surfaces. Another one was similar to Experiment 3, in which a planar grid of equally spaced points was defined. It iterated through the points in a linear order and changed their z-position according to the range rule, in relation to the already defined neighbouring points.

The experiments in this chapter show that it was not possible to create closed continuous surfaces while fixing the shape of an element and the angles in which this element proliferated. Also, changing the order of growth did not prevent the formation of gaps and overlaps. The only experiments that were successful were those in which the geometric elements were created in a three-dimensional reference network. This network was either a unit such as a box, as in Experiment 4 'Grow quads', or a three-dimensional matrix, as in Experiment 3 'Grow triangles' and Experiment 5 'Surfaces within a sphere network'.

CHAPTER | 6

‘UNIT CELL’

6

CHAPTER | 6

‘UNIT CELL’

6.1. Introduction

The Unit Cell is a geometric principle that allows families of shapes to be generated (Jonas 2014).

The features of the system are:

- the parts of one family in the Unit Cell growth model derive from the controlled variations of a polygonal shape
- the same parts can be assembled to form a diversity of global geometries
- these global geometries are gap-free articulated surfaces.

The Unit Cell itself is a three-dimensional parametric reference frame that is a part of a larger three-dimensional grid (Figure 100).

The main characteristic of the shapes which are created in a particular Unit Cell frame, is that the same parts can be assembled to form a wide range of different surface geometries (Figure 101). The rules controlling how the shapes connect are intrinsic in their geometry and defined by the edges along which the parts connect. An element can fit with copies of itself, or other elements and their rotations, if they share a compatible edge.

Each element in a shape family is unique and yet shares certain geometric features with the other members of the family. Shared features allow the parts to be connected to each other in different ways; features which distinguish them create the differentiation and articulation of the global surface geometries.

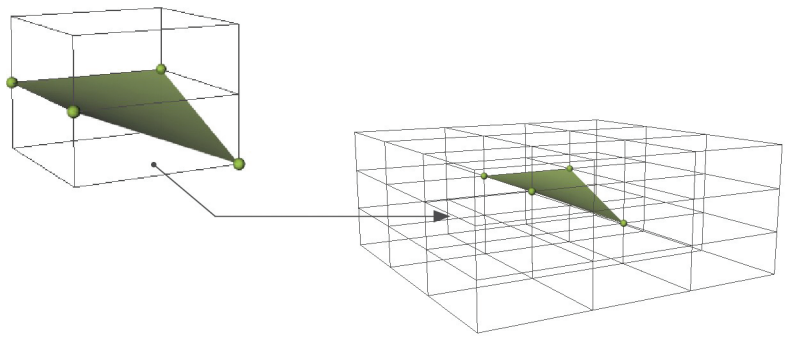


Figure 100
Unit Cell frame. Example of an element with a single lowered vertex (left), created in a simple box ‘Unit Cell’ which belongs to a larger orthogonal grid (right).

The key aspects of the Unit Cell are: the control of the number of elements which are created, and which are used for creating gap-free surface or frame structures; the recombinability of these elements, so that the same set of parts can form many global surface configurations, not just one. The system also considers the orthogonal physical offset of the parts, and through the explicit discreteness of their geometry, the transferability between different modelling and analysis applications. The Unit Cell principle is one possible answer to the dilemma described in the section on ‘Growing

form the naïve way' (page 38). The local geometric elements are controlled upfront, and yet the structures which evolve are not overlapping or branching out, but are continuous entities (unless the openings are desired). The Unit Cell was implemented as the growth model for the geometry system of the research project in this thesis.

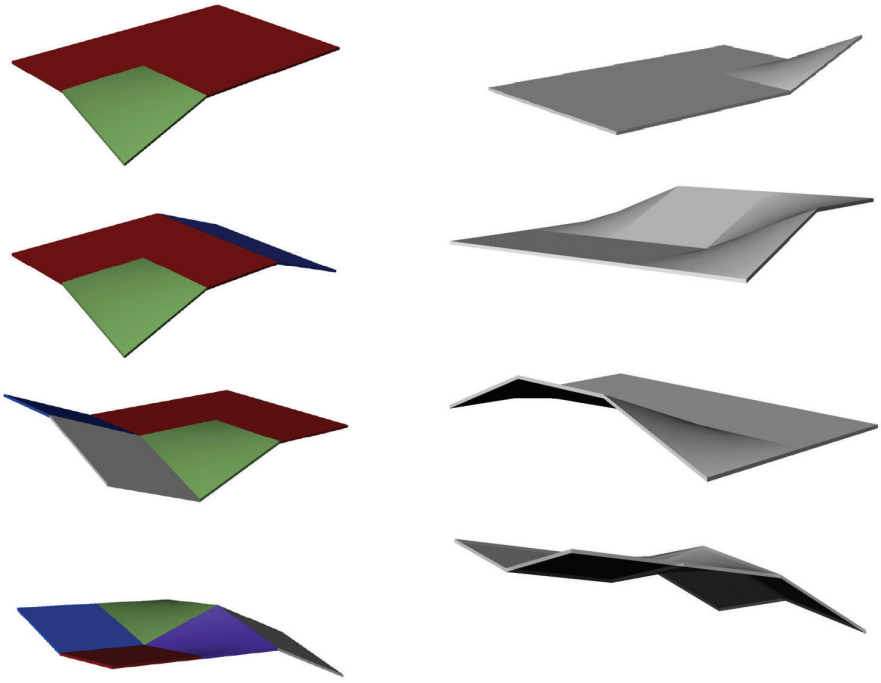


Figure 101
Surface configurations. Each consists of six elements and only six different tile forms. The surfaces on the left are colour coded. Each colour indicates a specific tile form. On the right, again, different articulations are shown, but the colour coding is turned off. These examples use a basic square-shape family of six different forms. The set of six forms is created in a square-based Unit Cell frame.

6.2. Representation

The Unit Cell system considers two representations of building components. The first representation is a surface or tile from which shell-type envelopes are created. The second representation is a node, thus a frame component, from which frame structures are created (Figure 102). Despite this important and differentiating specification, the Unit Cell growth model represents the building parts generically. This implies that they can be placeholders without the necessity of an explicit local form and material association. The distinction between surfaces and frames raises awareness of the fact that the same Unit Cell parameter settings result in different numbers of parts, depending on which representation is used. There are always more nodal forms in a frame family than tiles with the same Unit Cell parameter setting.

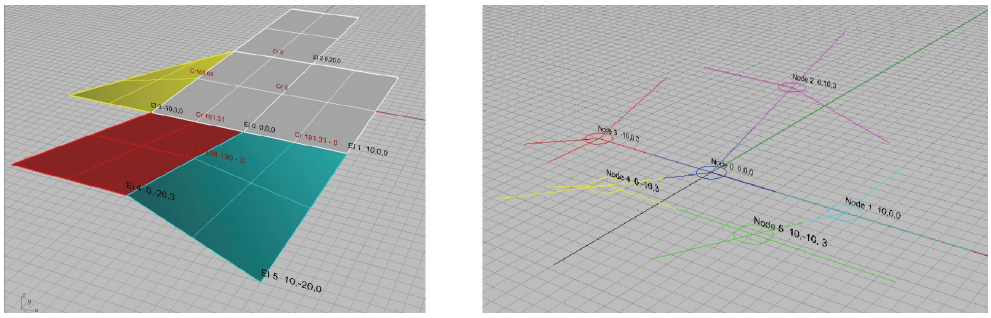


Figure 102
Representations. The system supports two different representations: tiles (surfaces) and frames (nodes).

6.3. Basic family of parts

A family, or set, of elements was created by going through all combinations of vertex displacement within a specific Unit Cell frame. In the simplest setup, the Unit Cell was a square frame in projection and had two levels (Figure 100). The distance between the central horizontal curve, and the upper and the lower curves was defined as one unit step. The starting point was four vertices in a plane, and the process went through all of the permutations of placing vertices one unit step up or down, or keeping them level. A rule which only allowed one unit step between the neighbouring vertices was applied. This produced six unique parts (Figure 103), one of them a flipped version of another. The form with only one vertex lifted was in principle the same as the one with one vertex lowered. Here, however, the two forms were distinctive, with respect to the generalisation of the representation of materials and types of connections. For example, if the parts represented a build-up of different layers of materials, then the form could not simply be flipped upside down. In projection, all elements exhibited a square footprint, which meant that the edges which were connected to a raised or lowered vertex were longer than those connected to a vertex with no displacement. In this particular example, the six forms showed two different types of edges, which were distinct through their orientation (horizontal or diagonal), the corresponding lengths and, if the edge was inclined, the direction. These characteristics determined whether one element fitted to another one in a particular rotation. In the implementation, the forms were indexed as vectors and sequenced in a particular order, so that if an edge was inclined, the position in the sequence and its prefix indicated the direction of the inclination. The production procedure was also applicable to a basic family of nodes.

The same process applied to the creation of a basic family of frames. In the simplest setup based on a square, the starting point was a central node with four rods coming out of it, one vertex at the end of each rod. At the beginning, the vertices lay in a plane, and the process went through all the permutations of placing them one unit up or down, or keeping them level. The same rule was applied as for the tile family, allowing only one unit step between the neighbouring vertices. This produced 24 unique frame elements (Figure 104).

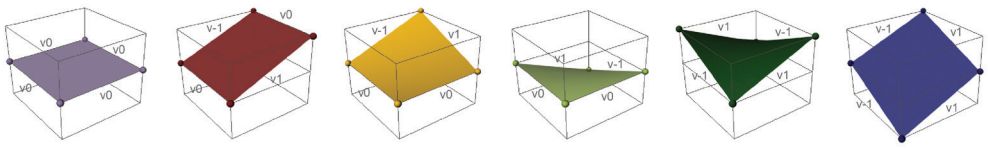


Figure 103
Basic tile family. This set of six unique elements referred to a square Unit Cell frame.

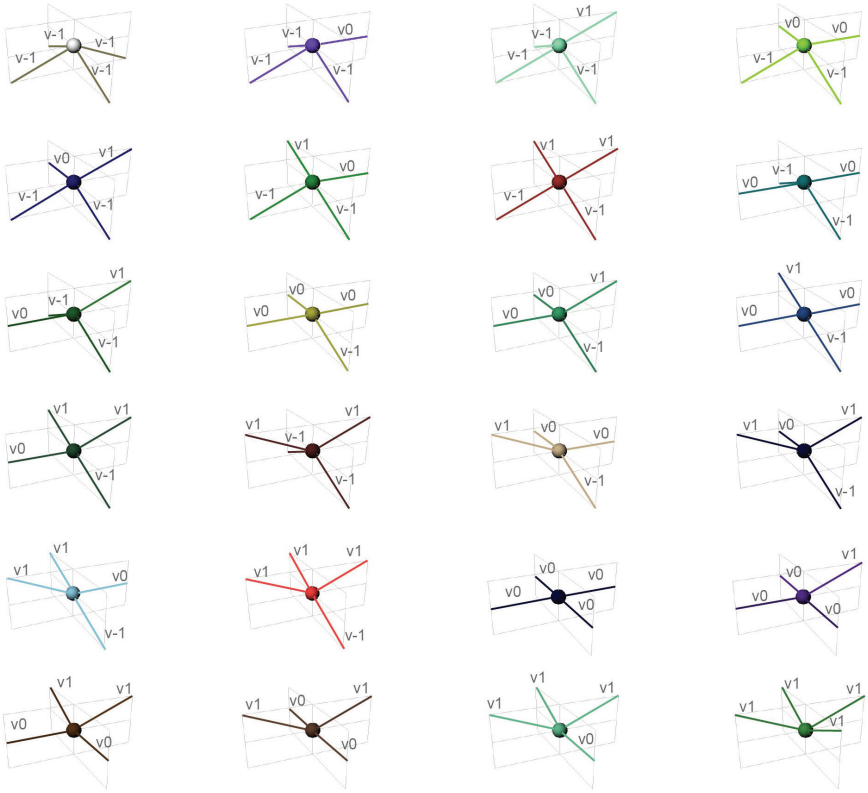


Figure 104
Example of the basic node family with 24 unique elements. This set of parts also referred to a square (in plan) Unit Cell frame.

6.4. Element factory - implementation

The Unit Cell concept works as an 'element factory'. The name indicates its utility as a custom element generator. It was implemented as a separate class to the larger plug-in, which was written in the C# language for the host CAD application, Rhinoceros. Elements are generated under user specifications at the beginning of each process run. It is an open generative engine which creates element families on the fly, in accordance with the user input. The element factory is an extendable parametric system, which was preferred over the idea of supplying a hard-coded, and thereby fixed, geometry set.

The parameters of the element factory are:

- Number of edges: the number of edges that each Unit Cell has. For example, a square tile element has four edges.
- Length of the edges: this parameter refers to the width and the length of the Unit Cell. In its current implementation, there is a restriction to the diversity of edge lengths. Facing edges are of the same length, but non-facing edges can be of different lengths. For instance, the square tile element has four edges and in projection all have the same length, but the rectangle has two different types of edge length.
- Height of the Unit Cell: the measure of one level step in the z-direction. The height of the Unit Cell affects the angle between the neighbouring components and thereby the length of that edge.

- Number of allowed steps in the z-direction (stepping value): the variable that indicates how many levels a vertex can go up or down relative to its neighbours, when iterating through the combinations of vertex positions. A stepping value of one, for example, specifies that the vertex can go up or down by one level.

Each setting produces a particular group of elements. To distinguish between the different types of elements, they are colour coded. Elements of the same type are indicated with the same colour.

The GUI of the prototype was limited to the choice between three- and four-sided elements, namely rectangles and triangles, with a minimum of one axis of symmetry being maintained. This was thought to be sufficient to test the concept and to ensure a robust prototype in the time available. The implemented code structure, akin to the Unit Cell concept, handles higher dimensions of polygonal shapes. The screenshots in Figure 105 show different shape families, the respective GUI settings and the corresponding sample surface.

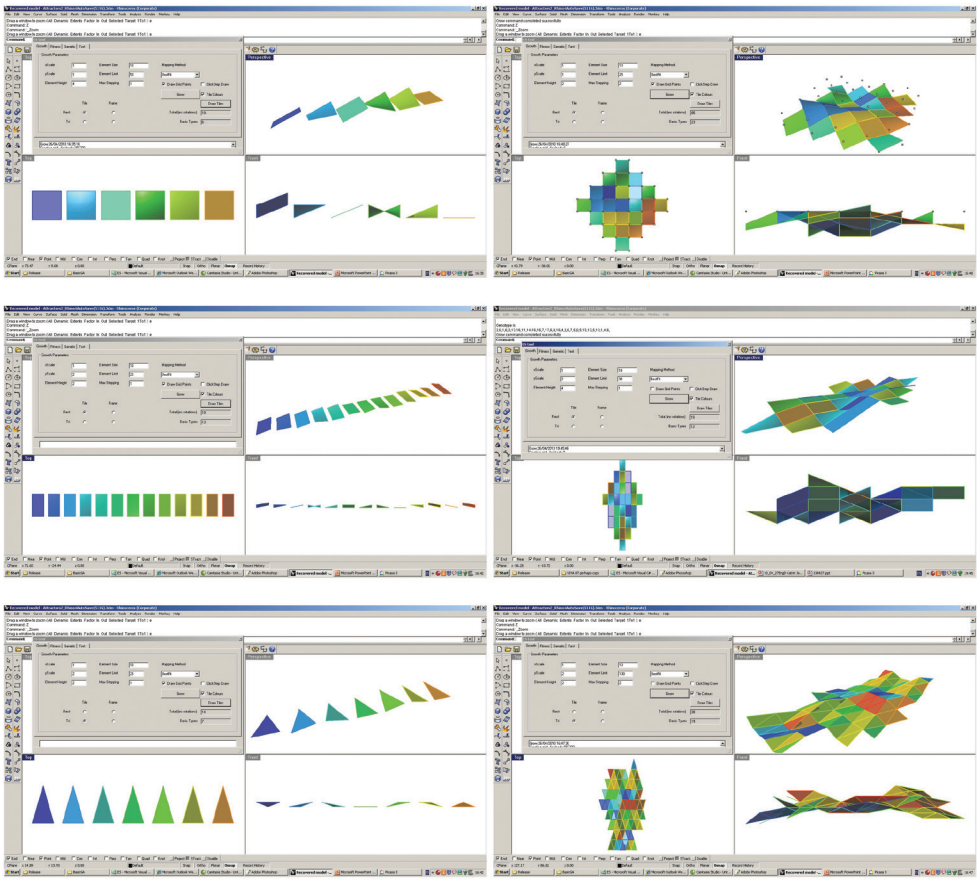


Figure 105
Rectangles and squares. The screenshots show basic changes in the shape parameters; from square to rectangle to triangle. Surface examples are on the right and the corresponding family of shapes on the left.

6.5. Combinatorics

The important feature of the Unit Cell growth model is the control of the number of elements which are then used for creating gap-free surface or frame structures (Figure 106). Three factors determine how many members a particular family of forms in the

Unit Cell growth model has. These three factors are the shape of the element, the stepping value and the representation.

Figure 107 and the table in Figure 110 show the influence of the first factor, the shape, when comparing the family of square tiles in the basic setup with the family of rectangular tiles. With the same growth parameter setting, the basic square family of elements has fewer members, six unique shapes, than the rectangular family, which has thirteen. The reason is that the rectangular Unit Cell frame has two different lengths of edges. When the elements are generated, the process has to apply a different filter to that for a square Unit Cell frame with two symmetry axes. Here, two types of rotation had to be considered and there is only a single axis of symmetry.

The second cause that increases the number of parts is the value for the allowed stepping in the z-direction. It is linked to the process of placing vertices up or down, or keeping them level. The number of possible combinations of vertex positions depends on the value of the level steps allowed. For instance, when two level steps are allowed, the procedure goes through all combinations of placing vertices one unit or two units up; the same is possible when the vertices are placed downwards, or they are kept at the same level. An example is shown in Figure 108 and the table in Figure 110.

The third factor is the representation of the elements, whether as surfaces or frames, as demonstrated in the screenshot in Figure 109 and the tables in Figure 111. There are always more nodes than connection angles and always more connection angles than elements in the case of the same growth parameter setting. The number of parts is reported instantly in the user interface when the user sets the parameters. This feedback helps the user to choose the parameter settings.

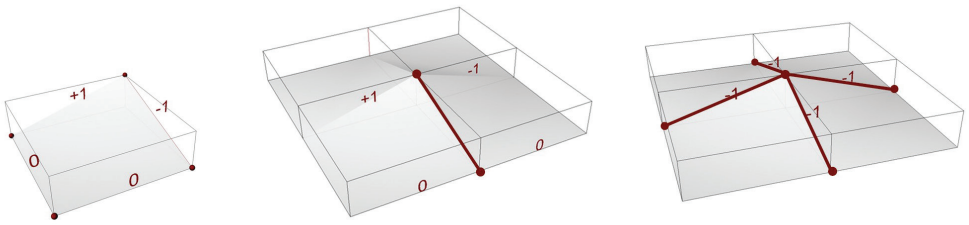


Figure 106
Number of element types. The Unit Cell method allows the user to control the number of element types used to build a larger surface geometry. These elements are local tiles (left), connections (middle) and nodes (right).

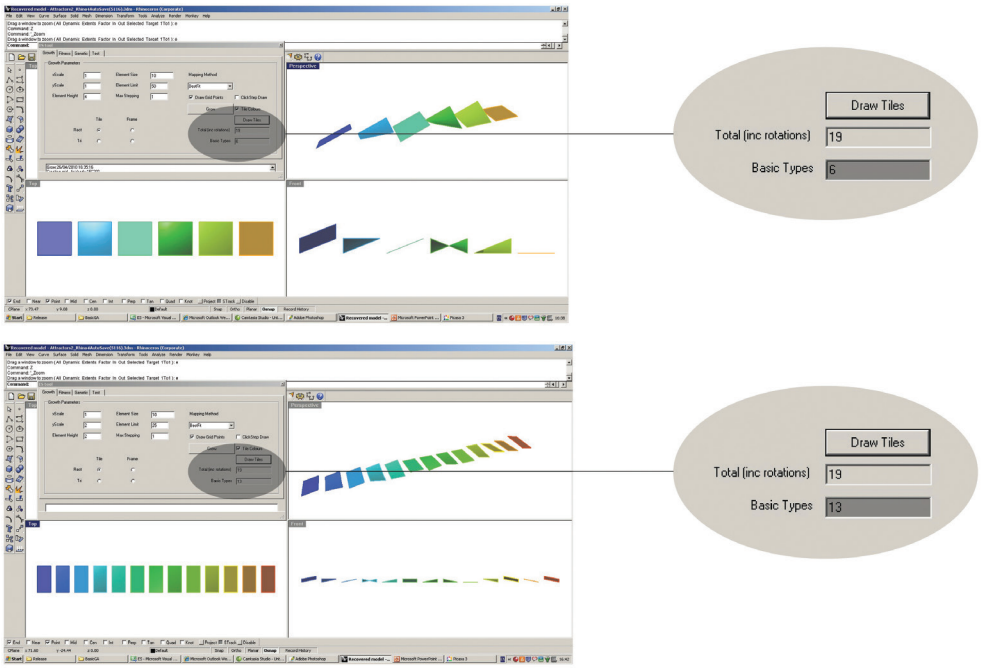


Figure 107
Different shapes. The basic square element family has six members. A small change to the setting, so that the generated shapes are rectangles, increases the size of the family to thirteen members.

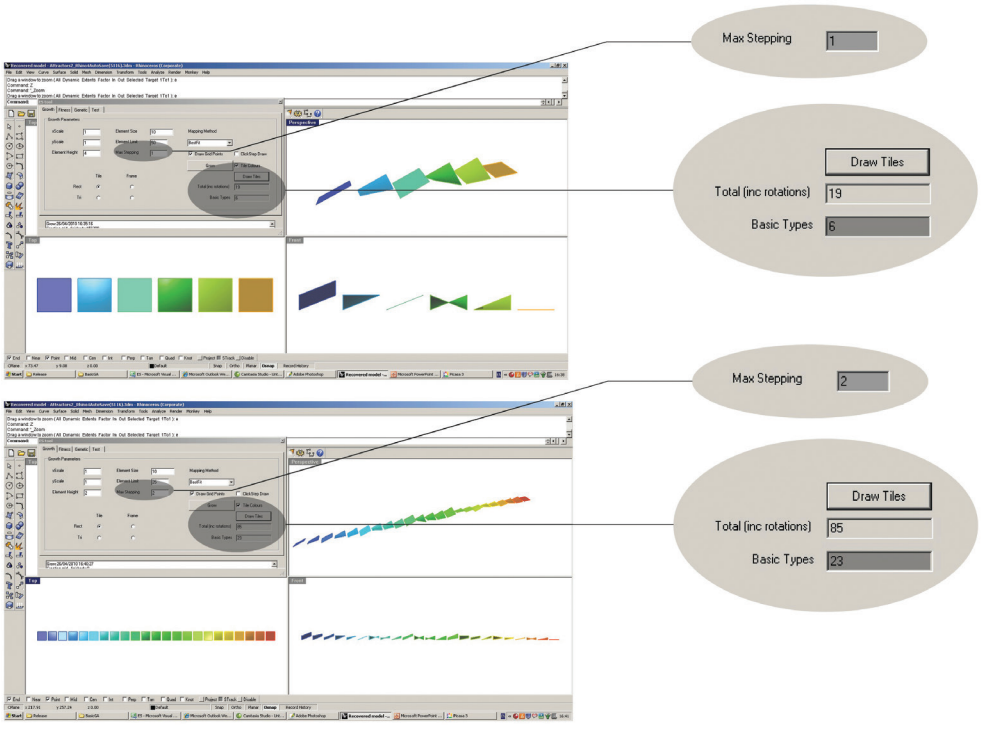
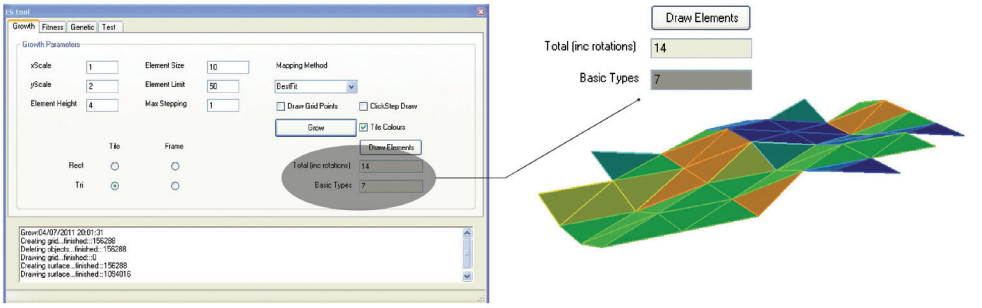


Figure 108
Different level stepping. The change in the allowed z-directional stepping affects the number of element types.



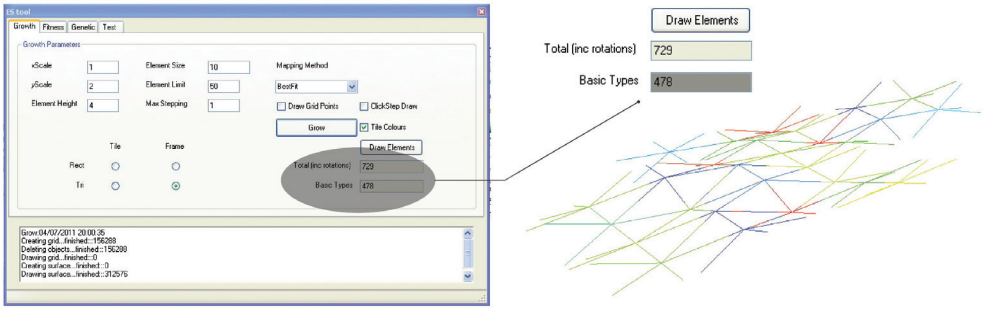


Figure 109
Surfaces or frames. The same surface configuration has many more node types than surface forms.

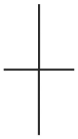
Shape representation in projection						
Tile	Node	Width	Length	Stepping	Number of tiles	Number of nodes
		1	1	1	6	24
		1	2	1	13	53
		1	1	1	3	130
		1	2	1	7	478

Figure 110
Shape comparison. The table shows the relationship between the projected shape of the Unit Cell, and the number of tile elements and frames.

Shape representation in projection		Width	Length	Stepping	Number of tiles
Tile	Node				
		1	1	1	6
		1	1	2	23
		1	1	3	60
		1	1	4	125

Figure 111
Stepping comparison. The table shows the relationship between the allowed level jumps in the z-direction, and the number of tile elements and frames.

6.5.1. Generating unique elements – process and notation

The element factory creates element families which were defined as sets. Every family is a set and every element in this family creates a subset.

Each element is described by a sequence of vectors (x,y,z). The x and y values of each element are the known parameters. They are defined through the parameter settings, wherein z is a variable. An element is described by providing a sequence of z values of its vectors, as x and y are already identified. Figure 112 shows the basic family of tiles of Figure 103 together with the notation of each element as a sequence of z-vectors ((z,ϵ[-s,s])z). The element factory generates a family set by going through all the possible vector combinations within the constraints. These constraints, expressed in the general notation, are n, the number of edges, and s, the number of level steppings. The vectors can go s levels up or down, or remain at the same level. A rule which only allows s level stepping between the neighbouring vertices is applicable here.

The vector strings are checked to see if they define valid elements (Figure 115). A description is valid if the sum of the z-vector values is 0. Valid elements are then compared, to filter out repetitions. Repetitions are rotations of the same vector sequence (Figure 116). Unique elements are recognised and their rotations are identified as types that have been defined already.

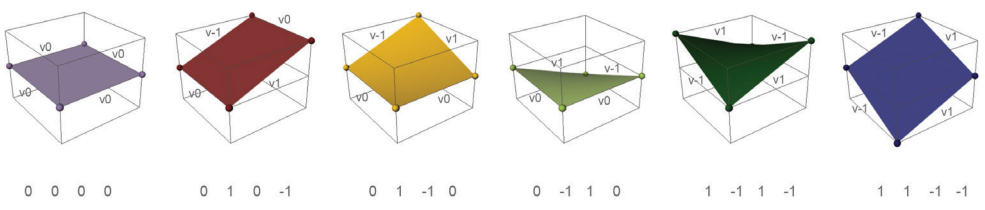
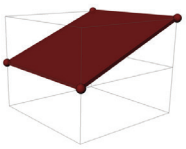


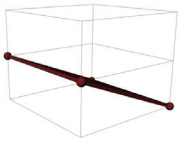
Figure 112
Basic tile family with vector notation. The basic family of tiles and the corresponding notation (clockwise), using z-value vectors.

Figure 113 shows an instance where the horizontal reflection (right) of the initial form (left) is equal to the rotation of the initial element. Therefore, the elements are the same. In contrast, Figure 114 shows two elements, where one is the horizontal reflection of the other. One cannot become the other by a simple rotation, therefore the two elements are distinct. The rotation and translations of an initial form are allowed, and counted as non-unique repetitions of the initial shape. Reflections are not considered to be repetitions.

Figure 113
Same element. Element on the right can be achieved either through rotation and translation of the tilted rectangle on the left, or through a horizontal reflection. The right element counts as the same type as the one on the left.

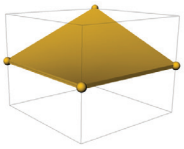


0 1 0 -1

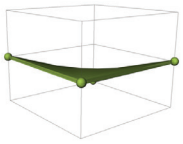


0 -1 0 1

Figure 114
Different element. Element on the right is a horizontal reflection of the element on the left. It counts as a different type of shape, as it cannot be achieved through rotation or translation of the tilted rectangle.



0 1 -1 0



0 -1 1 0

The procedure can be listed in three steps:

1. *Configure parameter settings.*
   The user inputs the parameters which determine the features of the elements (shape, representation, height of z-directional level jump and number of allowed level steps).

2. *Iterate through all possible vector combinations.*
   This process starts from a planar field of vectors and goes through all possible combinations of the vector configurations, considering the number of allowed level steps. Check whether it is a valid element; the z-vectors have to sum up to 0 (Figure 115).

3. *Iterate through all elements and find the unique ones.*
   This part of the procedure filters out those elements which are rotations of unique elements that have already been defined (Figure 116). If the element is not a rotation of an earlier defined element, then it is indicated as a new unique one. Reflections of a particular element count as new, unique ones. Colour coding is assigned, and unique elements and their respective rotations carry the same colour.

Figure 115
Valid element. Vector strings are checked for whether they define a valid element. A description is valid if the sum of the z-vector values is 0.

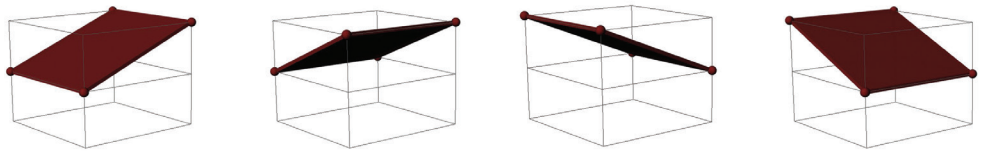


Figure 116
One element and its three rotations. All three rotations (second, third and fourth from the left) are the same as the initial element (far left).

The generation of a family set was implemented as a linear counting procedure and then as a grouping procedure. The number of parts is counted, at the same time as the vectors are stored which are necessary to actually draw the geometries. This combines the control of the number, as well as the geometrical definition of the elements.

The element factory program class was split into two separate streams, one for each representation: one to generate tiles and the other to generate nodes. This was an important decision, as it was fundamentally different to only changing the way in which the elements were displayed on the screen (Figure 117). This structure suggests a future extension to the program, where the user can pick individual elements from the preview and, in this way, decide which element enters the form-generating process.

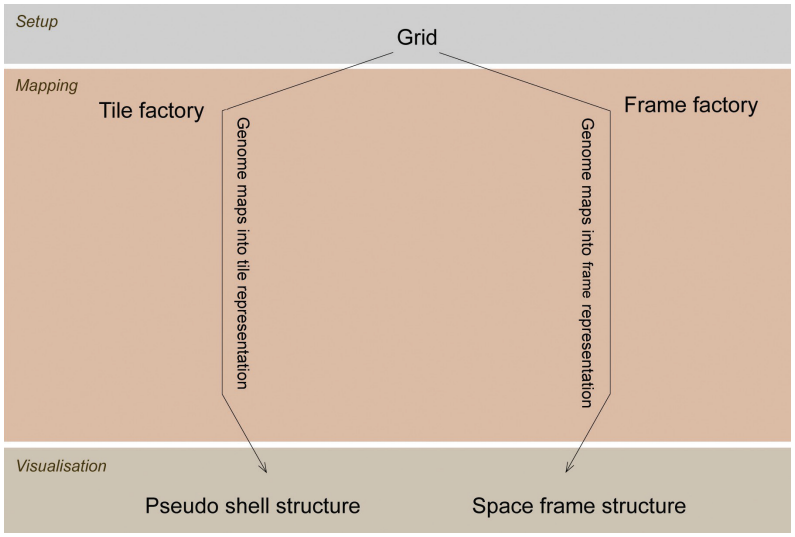


Figure 117
Two representations. Tiles and frames were programmed as separate strands in the element factory class.

6.5.2. Calculating number of parts – finding a general description

$$NT_n = 1 + \sum_{i=1}^{\lfloor \frac{n}{2} \rfloor} \frac{n!}{(n-2i)!(i!)^2}$$

Generalising the concept meant finding a formula which calculates the number of elements (including surface tiles, nodes and connections) for varying numbers of edges of the Unit Cell shape and level stepping. This task had to be broken down into manageable sub-problems. The first was to find the formula to calculate all elements for a stepping of 1 (-1, 0, 1), for all regular polygonal Unit Cell shapes with n edges. In this way, only the number of edges was generalised. The formula was derived from pairs of vector values, (-1/1) or ((-1, 1)/(1,-1)), which sum up to 0. 'NT' is the number of tiles, 'n' the number of edges and 'i' the iterator.

The calculation does not filter geometric repetitions. A commonly valid description of how to filter the elements is necessary to calculate the number of unique elements, excluding repetitions. The question is how to translate geometric criteria into a universal description. In the implementation, where every generated vector string was compared with previously and subsequently generated strings, the filtering of repeated elements was not a problem. However, a method of generalising this, and defining it as a formula for arbitrary polygonal shapes and varying numbers of level stepping, has yet to be explored. It is a recommendation of this thesis that further work is carried out to achieve this.

The n number of different parts was therefore calculated by using the method described previously and not by using an equation. This has not been a problem for the software; for a typical setup it takes less than a second to generate a family of forms and subsequently count the number of unique members in that family.

6.6. Summary

The growth model based on the Unit Cell principle allows the generation of discrete geometry sets. These sets are form families, of which each part can be connected to itself or to the other parts of the same family in various ways. Different larger surface geometries can be created using the same set of parts.

This geometry system provides a method to generate closed componential surfaces in a bottom-up process using discrete elements. It overcomes the dilemma of branching and overlap when trying to 'grow' a form in a 'naïve way'.

The Unit Cell is an architecturally integrated system. The control of the number of parts, the known physical offset, the interface to neighbouring parts (Figure 118) and the discreteness of the local geometries, which allows geometries to be easily transferred between the necessary software packages, informs the method using real-world parameters.

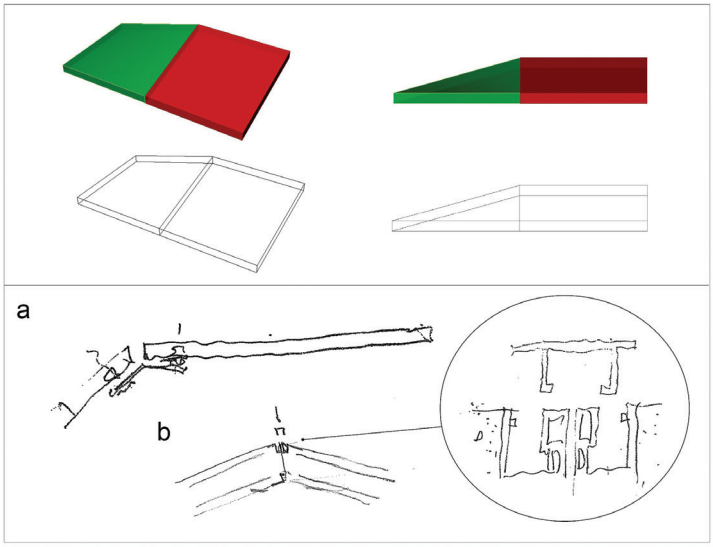


Figure 118
Interface with neighbouring parts. Example of two component forms that connect along one edge; isometric view (top left) and front view (top right). Sketch showing two possible connection types (bottom).

Elements which are defined within a three-dimensional Unit Cell frame, and therefore in a larger three-dimensional grid, carry the features of the underlying grid. The geometries defined within this grid will be repetitive if the matrix is regular, featuring repetitive cells. The term 'repetition' here involves repeated shapes and repeated angles between parts, and thereby connections and nodes. The more complex the grid, the more complex the geometry system defined within it. The grid can be associated with any coordinate system: the Cartesian coordinate system of the current implementation can, for example, be replaced by a polar coordinate system.

In summary, elements which are defined through the process of a combinational definition of vectors in a particular Unit Cell frame, such as an orthogonal box, create a family of forms. These forms carry shared features, the length and the shape of their edges, which allow them to be connected to each other. They also have distinct geometric characteristics through the different combinations of diagonally lifted, horizontal or lowered vectors. The distinction allows for diversity in global configurations when connecting the parts. Each set of parts created in a particular Unit Cell will create a catalogue of possible global geometry configurations.

The greater the repetition in the pattern of the global three-dimensional grid (the Unit Cell assembly), the greater the number of repeating elements which are defined within this grid. The principle of repetition can be maintained, if through spatial congruent transformations which are shape- and distance-preserving transformations (translation and rotation) (Pottmann 2010), one Unit Cell fits into different Unit Cells of the same cell assembly.

Comparison with ECSS tool

The Unit Cell principle overcomes the limitations of the fixed number of geometric parts that form the growth model of the ECSS tool, which is presented in the Review. The ECSS growth model is a hard-coded system, which uses similar intrinsic geometric rules to those of the Unit Cell. The elements of the ECSS can be compared to those of a simple square Unit Cell frame. The important difference is the shift from a limited set of discrete forms to a controlled set of parts. The Unit Cell is a parametric element generator which allows the user to manage the number of parts.

CHAPTER | 7

SURFACE APPROXIMATION

7

CHAPTER | 7

SURFACE APPROXIMATION

Surface approximation was a top-down experiment using the Unit Cell idea to approximate given surface geometries, as shown in Figure 119. The experiment served as an initial exploration of the limits and possibilities of the Unit Cell. It was possible to observe the types of global geometries that could be created, while trying out different polygonal shapes.

The general concept assumes that geometry sets can be defined in any regular or irregular three-dimensional grid that is defined in any coordinate system. This implementation of surface approximation looked at different grid patterns of layered two-dimensional grids, without any lateral translation in the third dimension or any other added complexity. The patterns of the grids used in this experiment were regular and the coordinate system that was used was the common Cartesian coordinate system.

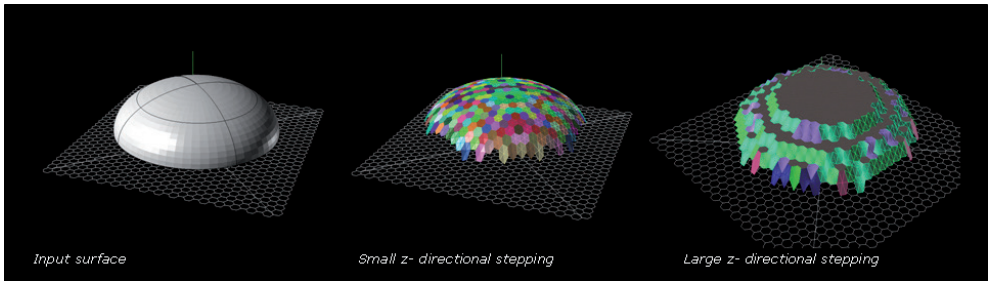


Figure 119
Dome shape-approximation. A dome-shaped surface was approximated using a hexagonal grid layout. Two z-directional level steps were tested: a small one (middle) and a large one (right).

The immediate purpose of the experiment was to observe the effects of the pattern of the grid, the size of its individual cells and the value of the level change on the resulting geometry. An example of changing cell sizes and level stepping in a simple orthogonal grid is shown in Figure 120.

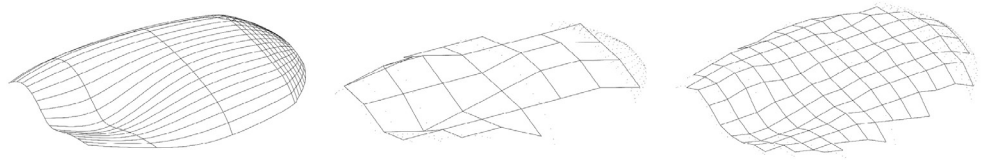


Figure 120
Freeform approximation. Input surface (left) and two approximations with different graining, i.e. cell size and height of level stepping (middle and right).

7.1. Process

The process adopts the idea of the Unit Cell by the following procedure. The user defines or selects a predefined planar grid of curves. The application asks the user to input a value for the z-level stepping, which automatically reads the two-dimensional grid pattern as a three-dimensional matrix with a level change size corresponding to the input value. An input surface is placed into the three-dimensional grid. The surface

is approximated by defining points at the surface and vertical grid-line intersection. The points still lie on the input surface, but then jump to the nearest node of the matrix. Points which outline elements are connected by closed curves and surfaces are created within these outlines. Depending on the underlying grid, the resulting surface is smoother or highly faceted in comparison with the input surface (Figure 121 and Figure 119). The elements are colour coded according to their surface area, to roughly sort them into types, assuming that every form has a unique area value associated with it (Figure 119). This means that elements which are flipped (upside-down) versions of each other will count as the same form, which is different to the Unit Cell implementations for the bottom-up growth of surfaces.

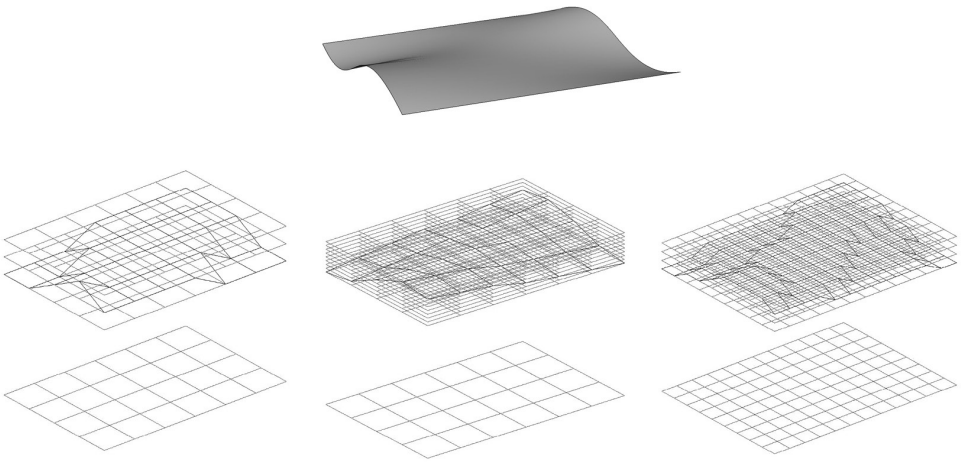


Figure 121
Approximation example. Approximation of an input surface with different changing z-directional level steps (left, middle and right) and changing cell size (right).

7.1.1. Implementation

Surface approximation is a plug-in for the host application Rhinoceros, written in C#.

The local elements are, in essence, the same as those defined in the Unit Cell growth model. However, the process is fundamentally different. Here, a surface point cloud adapts to a parametric grid by snapping its points to the nearest nodes in the grid. In contrast, the Unit Cell is essentially a bottom-up parametric kit of parts that is used to grow surfaces from scratch.

7.2. Experiments with different grid patterns

A series of tests was carried out, in which individual parameter alterations were made, while not changing the cell shape. The sequence of parameter alterations was then repeated with a different shape. The diagrams on the following pages document this study. In all instances, the input surface is the same doubly curved freeform.

Triangulated grid

The first two diagrams (Figure 122 and Figure 123) show the input surface approximated with triangular grids, each diagram comparing two different layouts and types of triangles. The first triangular layout uses equilateral triangles; the second uses right isosceles triangles. In the first layout, the triangles create a somewhat

homogenously spread pattern. There are horizontal straight lines and diagonal lines in two mirrored directions. In the second layout, the pattern is oriented dominantly in one direction, because the diagonal lines which add to the vertical division run only in one direction. The effect of the type of triangle and orientation on the outcome is more easily viewed in the larger scale triangulation (Figure 123). The number of different element types created, in relation to the total number of parts, is similar in both surfaces, despite the different triangulation patterns. In the small tessellation with the equilateral triangulation, there are 33 unique parts of 787 elements in total. In the bigger equilateral tessellation, 30 of a total of 49 elements are unique. In the small tessellation with the isosceles triangulation pattern, there are 35 unique elements of a total of 798 parts, and in the bigger tessellation, 40 elements of 60 are unique.

Hexagonal grid

Figure 124 shows the input surface approximated by a hexagonal grid with small tile units in the upper row and another one with large tile units in the lower row. Hexagons can be viewed as clusters of triangles, akin to the equilateral triangle tessellation shown in Figure 122 and Figure 123. The ratio of unique elements to elements in total is 71 to 1171 for the small cell size tessellation and 18 to 23 for the approximation with the bigger cell size.

Square grid

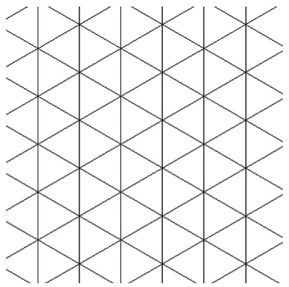
The diagram illustrating trials with the square grid (Figure 125) focuses on a change in the height of one unit step in the z-direction. The surface shown in the upper row is created using a relatively small unit step in the z-direction. As was to be expected, the surface outcome is smoother than the surface shown in the second row, which has a much coarser appearance and which has the corresponding larger size of the z-directional unit step. The program does not include a control mechanism to restrict the number of level steps. Therefore, if the unit step in the z-direction is small, then it is likely that the neighbouring points are separated by multiple level jumps. The outcome is then closer to the input surface, but there are many shapes created. The surface consists of 895 elements in all and 64 different areas are recognised.

In the coarser approximation, with a large z-directional step height, the neighbouring points are fewer level steps apart from each other. Therefore, there are fewer element types created than in the finer approximation. It also has 895 tiles in all, but only 17 different types.

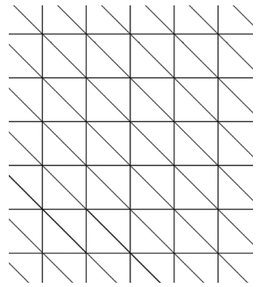
Rotated star grid

The rotated star grid (Figure 126) has a complex appearance. This does not automatically mean that more tile forms are needed to approximate the input surface than in a visually less elaborate grid. Figure 126 compares two different settings for the heights of the unit step in the z-direction; this is the same as the square grid diagram shown in Figure 125. The small height for the z-directional step in the surface shown in the upper row produces a smooth surface. It is built from 1534 tiles in total and exhibits 67 different tile types. The rougher approximation, where a larger unit step in z-direction is used, consists of 20 unique shapes.

In this diagram two grids with different triangulation patterns are compared. The first approximation (upper sequence) shows a surface built up from small equilateral triangles. The surface has 787 elements in total and 33 different tile forms. The second approximation is built up from small isosceles triangles. The surface has 789 elements in total and 35 different tile forms. For both surfaces a small stepping level was used, which causes a smoother surface, yet a high number of different element types.

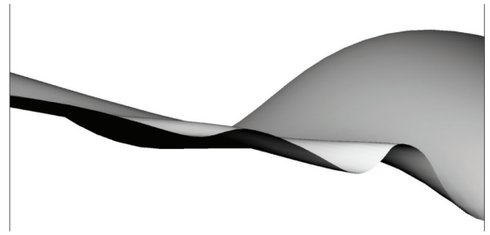


Equilateral triangulation, grid type A



Isosceles triangulation, grid type B

Close-up of input surface



Surface properties

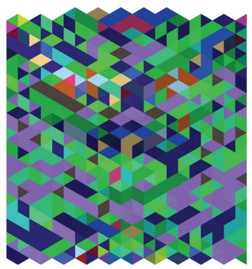
Plan view

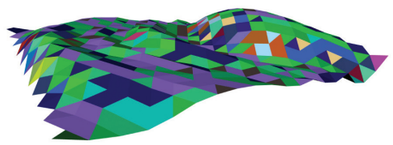
Isometric view

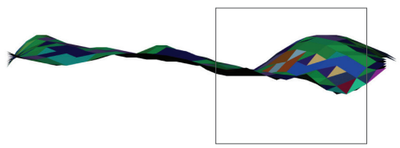
Elevation

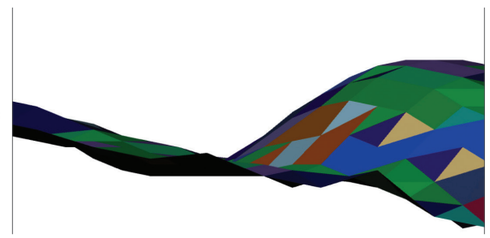
Close-up of approximated surface

Index	33/787
Altered parameter	Triangulation type A
Grid type/ Tessellation pattern	Equilateral triangles, small
Z- directional level stepping	Small, 3 units
Number of elements in the surface	787
Number of different element types	33

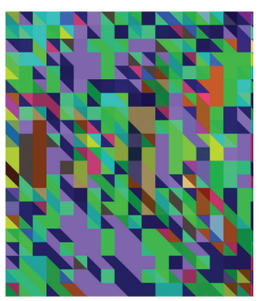


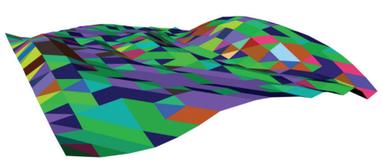


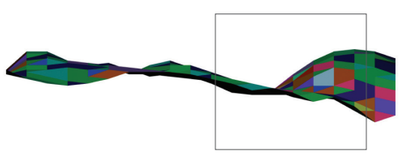




Index	35/98
Altered parameter	Triangulation type B
Grid type/ Tessellation pattern	Isosceles triangles, small
Z- directional level stepping	Small, 3 units
Number of elements in the surface	798
Number of different element types	35







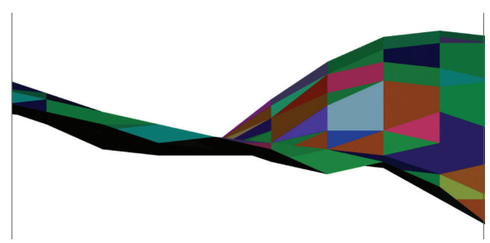
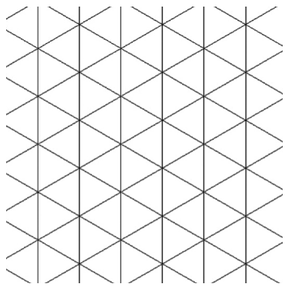
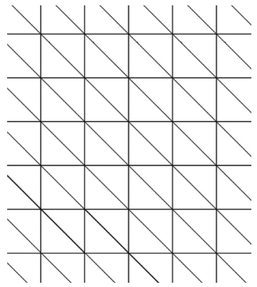


Figure 122
Triangulated grids, small tiling. Different pattern of triangulation.

In this diagram the two different triangulation patterns are compared, with a large cell size. The upper sequence shows a surface built up from big equilateral triangles. The surface has 49 elements in total and 30 different tile forms. The second sequence, in the lower row, is built up from big isosceles triangles. The surface has 60 elements in total and 40 different tile forms. For both surfaces a small stepping level was used.

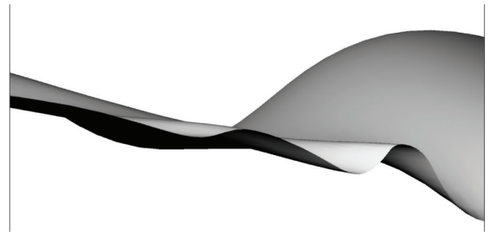


Equilateral triangulation, grid type A



Isosceles triangulation, grid type B

Close-up of input surface

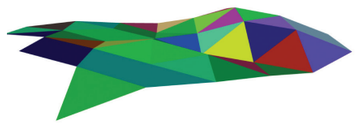


Surface properties

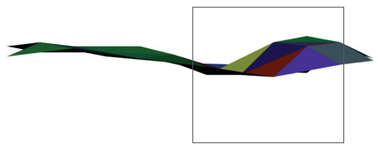
Index	30/497
Altered parameter	Triangulation type A
Grid type/ Tessellation pattern	Equilateral triangles, big
Z- directional level stepping	Small, 3 units
Number of elements in the surface	49
Number of different element types	30

Plan view

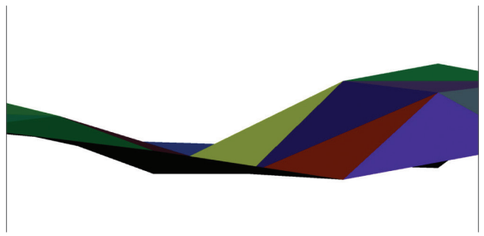
Isometric view



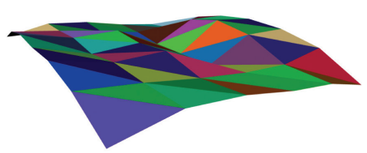
Elevation

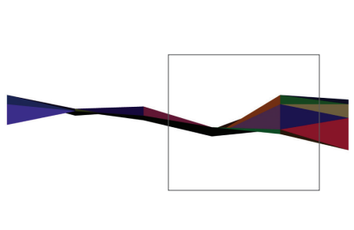


Close-up of approximated surface



Index	40/60
Altered parameter	Triangulation type B
Grid type/ Tessellation pattern	Isosceles triangles, big
Z- directional level stepping	Small, 3 units
Number of elements in the surface	60
Number of different element types	30





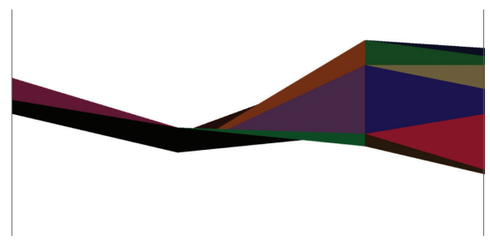
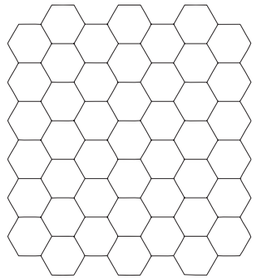
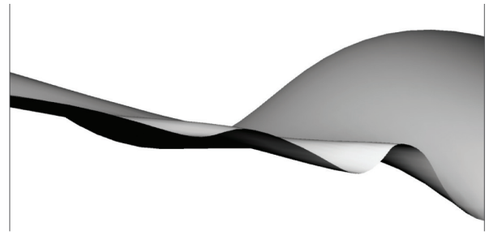


Figure 123
Triangulated grids, big tiling. Different pattern of triangulation.

The approximation in this diagram shows two grids with different sizes of the individual cell. The tessellation pattern is hexagonal. The first approximation (upper sequence) is built up from small hexagonal tiles. The surface has 1171 elements in total and 71 different tile forms. The second approximation is built up from bigger hexagonal tiles. The surface has 23 elements in total and 18 different tile forms. In both surface approximations a small stepping level was used, which caused a smoother surface, yet a high number of different element types.



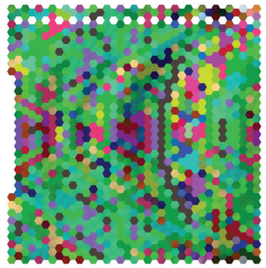
Close-up of input surface



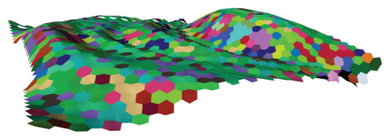
Surface properties

Index	71/171
Altered parameter	Small shape
Grid type/ Tessellation pattern	Hexagonal
Z- directional level stepping	Small, 3 units
Number of elements in the surface	1171
Number of different element types	71

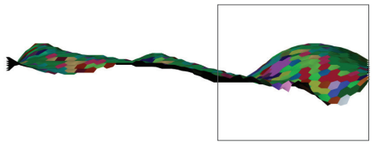
Plan view



Isometric view

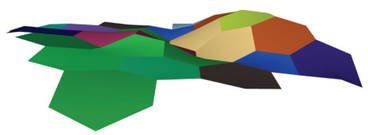


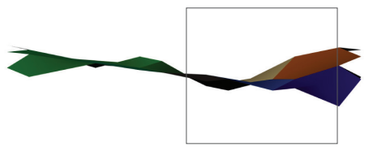
Elevation



Close-up of approximated surface

Index	18/23
Altered parameter	Big shape
Grid type/ Tessellation pattern	Hexagonal
Z- directional level stepping	Small, 3 units
Number of elements in the surface	23
Number of different element types	18





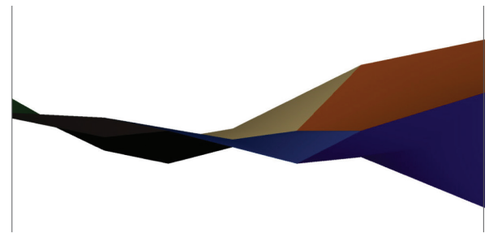
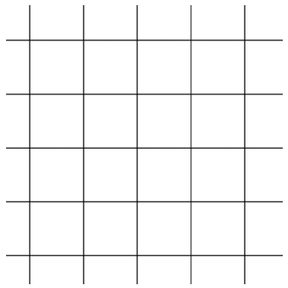
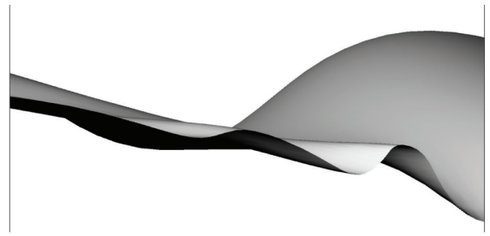


Figure 124
Hexagonal grid. Different size of tiling.

This diagram shows two approximations, each with a different z-level stepping and simple squared tessellation pattern. The surface shown in the upper sequence has a small z-level stepping and produces a smooth description of the input surface. It has 895 tile elements in total and needs 64 unique tile forms. The second approximation uses a bigger level stepping. It has the same number of elements but needs only 17 unique forms.



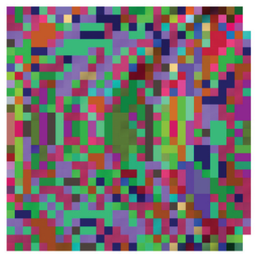
Close-up of input surface



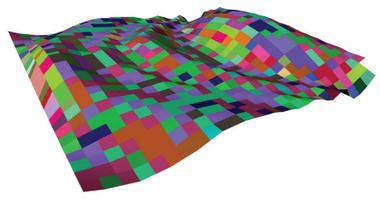
Surface properties

Index	3/b/3 64/895
Altered parameter	Small level stepping
Grid type/ Tessellation pattern	Square
Z- directional level stepping	Small, 3 units
Number of elements in the surface	895
Number of different element types	64

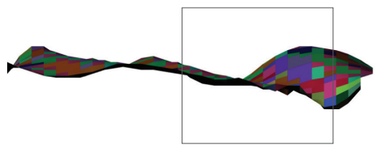
Plan view



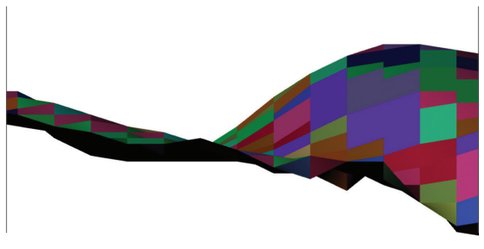
Isometric view



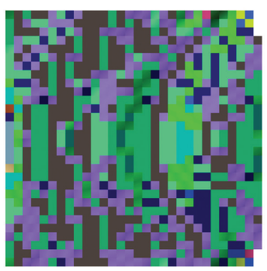
Elevation

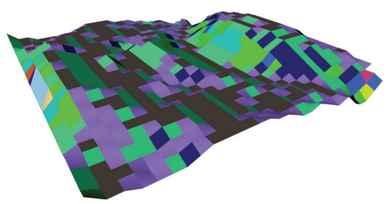


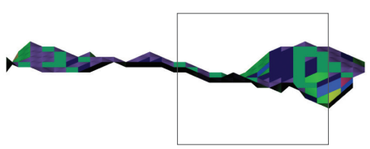
Close-up of approximated surface



Index	3/b/8 17/895
Altered parameter	Big level stepping
Grid type/ Tessellation pattern	Square
Z- directional level stepping	Small, 3 units
Number of elements in the surface	895
Number of different element types	17







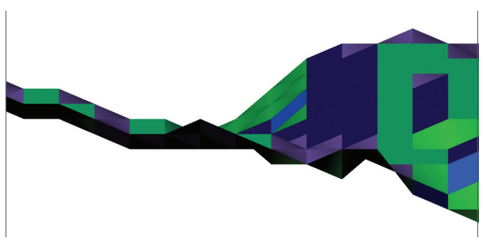
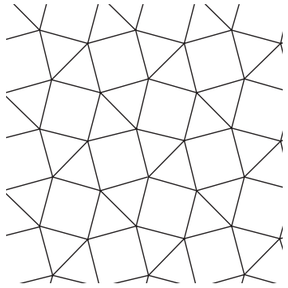
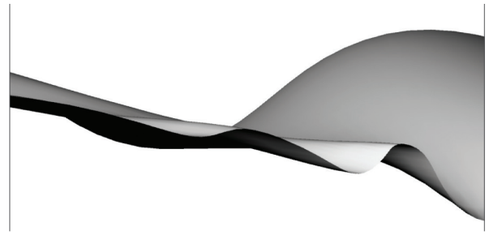


Figure 125
Square grid. Different level stepping.

This diagram shows two approximations, each with a different z-level stepping. Despite the elaborate appearance of the tessellation it has a repetitive pattern and therefore does not need more tile forms than visually simpler tessellations. The surface shown in the upper sequence has a small z-level stepping and produces a fairly smooth articulation. It has 1534 tile elements in total and needs 67 unique tile forms. The second approximation uses a bigger level stepping. It has the same number of elements but needs only 20 unique forms.



Close-up of input surface



Surface properties

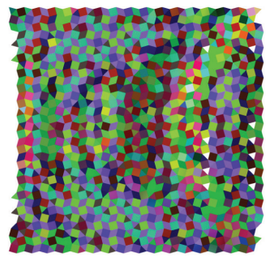
Plan view

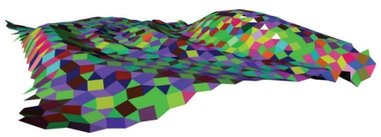
Isometric view

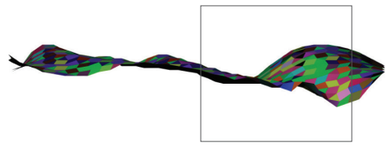
Elevation

Close-up of approximated surface

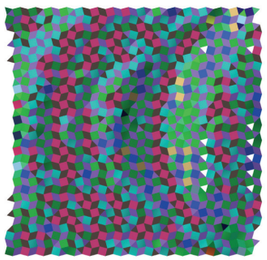
Index	5/b/3 67/1534
Altered parameter	Small level stepping
Grid type/ Tessellation pattern	Rotated star
Z- directional level stepping	Small, 3 units
Number of elements in the surface	1534
Number of different element types	67

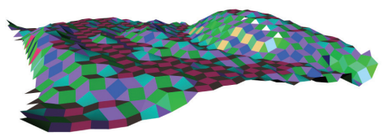


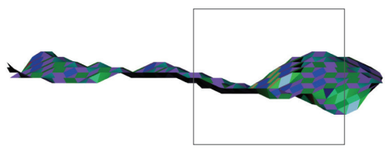




Index	5/b/8 20/1534
Altered parameter	Big level stepping
Grid type/ Tessellation pattern	Rotated star
Z- directional level stepping	Big, 8 units
Number of elements in the surface	1534
Number of different element types	20







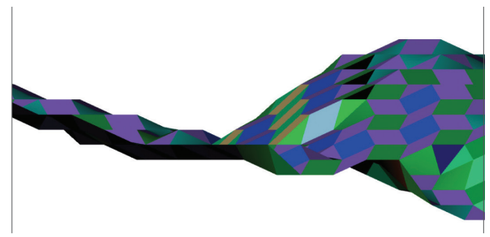


Figure 126
Rotated grid.
Different level stepping.

7.3. Conclusion on the surface approximation experiment

The top-down surface approximation experiment showed that when using this system the known surface types such as a regular dome, as well as freeforms, could be approximated with a custom pattern of subdivision, while controlling the number of employed element types. The documented approximations only used grids with homogeneous patterns. However, surfaces can be tessellated into any regular or irregular pattern, using this process. The restriction of the regular pattern corresponds to the current setup of the Unit Cell, where only the z-level stepping is a variable.

Concentrating on the relationship between the parameters which governed the procedure, namely the cell size and the number of allowed level steps, a number of observations could be made. Smooth articulations could be adopted if the cell size is small (value of one unit step in the x, y and z dimensions), while the number of allowed level steps is big (vertices can jump more than one level up or down). This is particularly so for forms with steep curvature, or sudden changes in the steepness or direction of articulation. If the correlation between cell size and level steps exists as described above, it produces a fine graining of the tessellation and thereby allows a greater adaptation to the input surface. If multiple level steps are allowed, the number of necessary form types increases, due to the different angles between the parts. The smaller the tessellation, the higher the total number of necessary connections. Both the number of different types and the total number of connections are vital cost factors. To achieve a close match to the input surface, while at the same time managing the cost, it is desirable for the approximation tool to include mixed sizes of cells. This implies additional stepping, akin to the z-directional level stepping, but in the x- and y-directions, which could become an extension to the plug-in.

The top-down experiment showed that the Unit Cell growth model in a bottom-up process could be extended to include polygonal shapes with more sides. This is interesting for structures where the tessellation pattern becomes an active part of the design. The study also showed that, even when the employed geometric parts are highly constrained, different surface types can be created. However, if greater flexibility is desired for detailed articulation, additional stepping in the x- and y-dimensions may be useful. One example would be where there is a requirement to achieve a smooth transition from a steep-to-shallow curvature and directional changes in the articulation, while keeping the number of component types low.

CHAPTER | 8

FORM GROWTH

8

CHAPTER | 8

FORM GROWTH

Form growth is the inductive generation of form, where smaller parts are proliferated to create larger geometric entities, surfaces in this case. The growth process takes in the growth model, the growth language and the growth order. The growth model consists of the geometries and the rules of how to connect them. The growth language is the symbolic representation of the growth model, to allow for a fast computational process. The growth order is the sequence in which the components are placed.

The previous chapters presented the development of the Unit Cell growth model and a verification study of the process. In this chapter, the Unit Cell is integrated into a form-generating system. To incorporate the element factory class of the growth model, the surface growth is implemented in C# for the host application Rhinoceros.

8.1. Process

In this implementation, the growth of a surface starts by default with a randomly selected element, the seed, at the centre of the growth field. The growth could start at any point in the growth field and with a specific element, if that is desired. The growth field is the underlying lattice. The size of this lattice is defined by the user inputting the surface size. According to the growth order, subsequent components are placed in a specified sequence. The rules of the growth operate locally, meaning the last placed component constrains the choice for the next components that can be placed with respect to the growth model.

The choice of which element is placed next depends on which element can fit. Due to the elements placed already, there is only a subset of the family of parts which can fit into a particular position. Which of this subset is chosen can be either determined randomly or through a decision mechanism. One such decision mechanism is the integer string (genome) in an evolutionary optimisation cycle, where the number string, together with the growth model and the growth order, ensures that a particular surface is recreated. All elements that belong to the respective generated family of parts are involved in the generation process. At this stage, the user cannot make a sub-selection of elements.

8.2. Growth parameters

Growth parameters are user-controlled constraints for the local geometry sets (Figure 127). They specify the properties of the underlying Unit Cell, and thereby the shape reference and size reference of the local parts, as well as the dimension of the final surfaces.

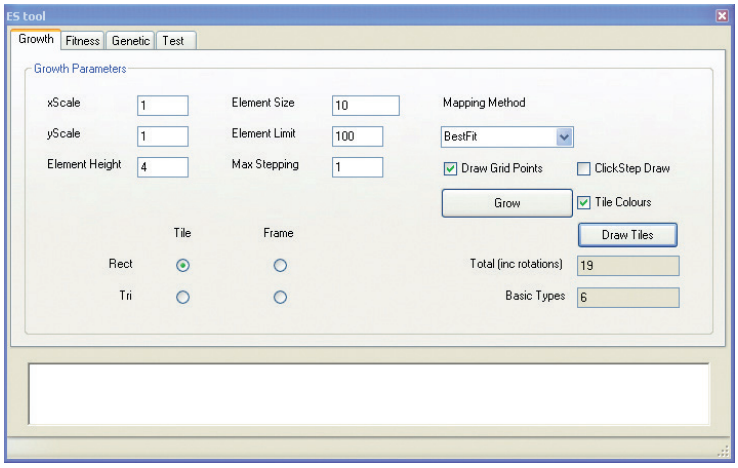


Figure 127
Graphical user interface for the growth parameters.

8.2.1. Growth order

One important parameter in the growth process, although not a user control parameter in this setup, is the growth order. The growth order is the path along which the elements are placed. The aspiration for the growth order in the design of the tool discussed here was twofold. Firstly, it was to allow the surfaces to develop with the greatest possible freedom, so that the range of the probable global configurations was as wide as possible. The size of the range is controlled by the number of choices at each position that has to be filled in the growth sequence. This in turn depends on how many neighbours of this position are placed already, as these predefine the element parts in which they connect to the next element; the edges of a tile act as rods of a frame. There is a hierarchy of dependency in every combination of the growth model and the growth order.

The second aim was to ensure that the surfaces were continuous entities which did not branch out, unless that was a desired feature. This depended on whether or not local positions could be filled with an element that fitted, before the growth continued outwards. Different growth orders were investigated, always starting from a central seed element. The following dependency diagrams show the different growth order options considered: linear, random, square, circular and rhomboidal (Figure 128). The colour coding indicates the respective level of freedom to fill a position (Figure 129, Figure 130, Figure 131, Figure 132, and Figure 133). Green signifies the maximum freedom, including the initial seed element and the positions where only one rod is defined. Blue denotes restricted freedom, where two of the four rods are already determined by the neighbouring elements.

The linear growth order was studied in 'Experiment 3 – Grow triangles in a grid' (5.7). It showed that in a linear growth sequence, the first row dictated the articulation of the rest of the surface and it became fixed in one direction. This is illustrated in the dependency diagram presented in Figure 129 where only the first row has a single

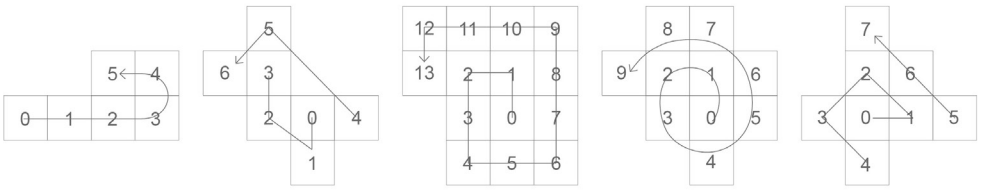


Figure 128
Considered and tested growth orders. The illustration shows the different growth orders: linear (left), random (second from left), square (third from left), circular (fourth from left) and rhomboidal (right).

constraint, in that one edge is defined already. Elements in the following rows are constrained through their neighbours in the same and the previous rows; therefore two edges are already defined. The linear order created closed surfaces, but did not seem sufficiently flexible to allow enough freedom for the development of the articulation of the surfaces.

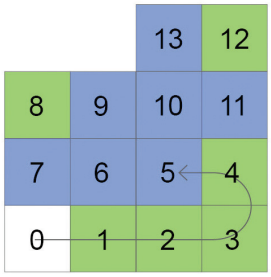


Figure 129
Linear growth order. Illustration of the linear growth order as a dependency diagram. Positions with a single predefined edge are indicated in green; those with two predefined edges are indicated in blue.

The random growth order first appeared suitable, as it offered the greatest freedom for the development of the global form. Figure 130 shows that the colour of most positions is green, which indicates that only a single edge of the tile to be placed next is determined through a neighbouring tile. However, surfaces could not be recreated. In the later implementation of the geometry system in the optimisation process, a prerequisite was that the surfaces could be redrawn given their genome (symbolic representation) and the order in which they were placed. This was not possible with the random growth order. Another drawback was that the structures branched out and overlapped, or created gaps. The possibility of the latter is illustrated in Figure 130. Position 13 is circumscribed by tiles and thereby completely determined. If none of the available tile types fits this prescription, the position remains empty. Additional rules could have been introduced to stop this from happening, but because of the first drawback mentioned, the random growth order was not put forward.

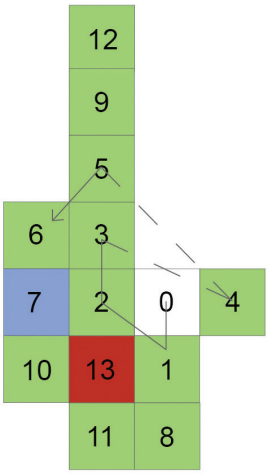


Figure 130
Random growth order. Positions with a single predefined edge are green, the one with two edges defined is blue and the one where all edges are already determined is red.

Both the square growth order (Figure 131) and circular growth order (Figure 132) exhibit a similar ratio of single and double constraint positions to the linear organisation. The diagram for the rhomboidal order (Figure 133) displays a higher degree of freedom, similar to the random growth sequence. It allows the reproduction of surfaces and the creation of gap-free structures. It was therefore chosen as the fixed order for the growth.

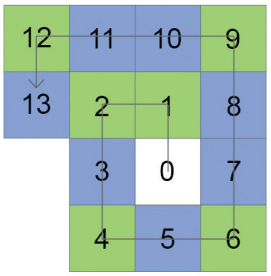


Figure 131
Square growth order. Positions with a single predefined edge are green; those with two predefined edges are blue.

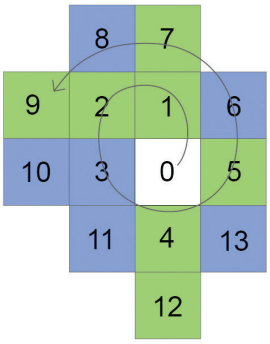


Figure 132
Circular growth order. Positions with a single predefined edge are green; those with two predefined edges are blue.

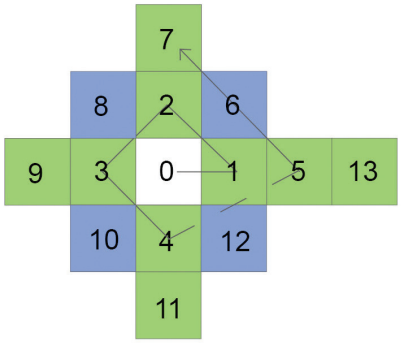


Figure 133
Rhomboidal growth order. Positions with a single predefined edge are green; those with two predefined edges are blue.

8.2.2. Visualising dependencies

The diagrams in the previous paragraph were manually derived, and focussed on the tile representation and square element shape. An algorithmic routine was developed to generate the dependency diagrams automatically, combining different shapes and growth orders, this time for the frame representation. The procedure created maps by colour coding and numbering the degree of choice at each position in the growth. The routine demonstrated that the dependency layout for the same growth order changed if the grid was altered and vice versa (Figure 134). The routine was a good way to find a generally suitable order for every shape family or a particular organisation for each individual shape family, and thereby grid layout. The exercise suggested that greater investigations could be carried out to find suitable growth orders for specific shapes.

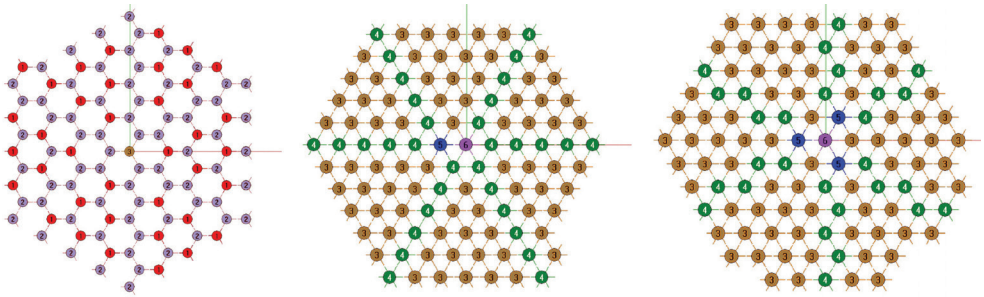


Figure 134
Dependency maps. The generated diagrams show the dependency for a hexagonal grid with a circular growth order (left), a triangular grid with a circular growth order (middle) and a triangular grid with a rhomboidal growth order (right).

8.3. Mapping indexes

The challenge in the implementation of the growth was to generalise the rules of the process and to integrate the parametric growth model, with its different shapes and sizes, and two representations.

To ensure that the bottom-up growth worked for all growth model settings, the indexing topology had to work for all scenarios. This required a mapping structure, where different levels of indexes and coordinate systems referred to each other. The starting point was the regular orthogonal coordinate system for the grid and the grid points, to specify the growth within the global three-dimensional matrix. This translated into the rhomboidal coordinate system, which directed the growth. It sat at the starting point of the growth at the centre of the lattice.

Then there were the nodes, which became the vertices of the shapes and the vectors. At the beginning of each growth process, the projected x- and y-description of each surface was known; the articulation in the z-dimension was generated during the process. The mapping ensured that the lattice, the growth order and the growth model were referenced to each other. The growth order and the growth model index are shown in Figure 135.

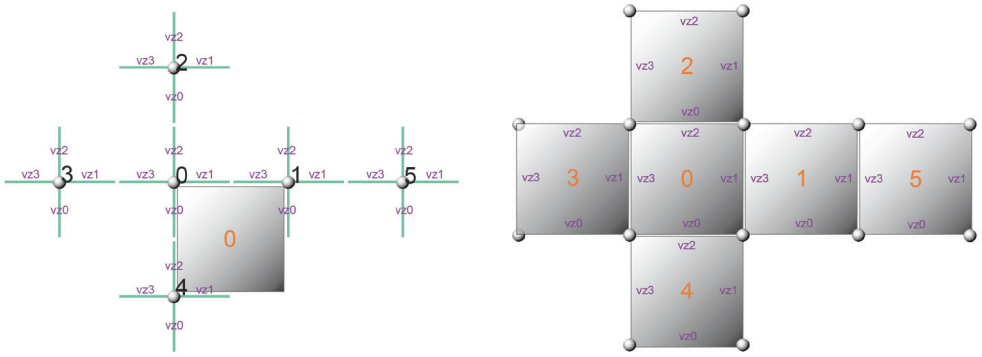


Figure 135
Indexing. Illustration of the index map for a frame growth in a square grid, showing the index for the nodes (spheres, and big black and blue numbers), the node rods (green numbers), the rod vectors (pink variable), the field (orange) and the edges of the field (light orange), which later also refer to the edge vector variable (not shown).

8.4. Summary

The algorithmic form growth to generate componential surfaces can be used as an independent design tool. The use of the principle of the Unit Cell as the growth model supported the idea of designing a system of simple repetitive parts and rules, defining how they interact to create larger structures with a higher level of complexity. All the resulting surfaces are buildable and, despite the constrained geometry, the catalogue of possible outcomes is extensive.

The user controls the parameters for the growth, the shape, the representation and the stepping of the components. The growth order was fixed to the rhomboidal growth sequence, as it allowed for the greatest freedom, while ensuring that the surfaces could be recreated.

The way in which the growth model was implemented, particularly the indexing to organise the growth, allowed the inclusion of the different shapes and two representations. It also prepared the system to be extended to include shapes with more or different numbers of sides, as was suggested by the approximation experiments.

CHAPTER | 9

SEARCH AND OPTIMISATION

9

CHAPTER | 9

SEARCH AND OPTIMISATION

9.1. Extending the growth process with an evolutionary search mechanism

If the number of combinatory outcomes in an inductive generative process is large, then it needs a selection mechanism. In the presented tool, the number of possible surfaces is high. Therefore, there are two levels of selection control. The surface generation engine becomes the seed for an evolutionary algorithm (EA). The EA allows searching this space in an efficient way to find solutions and optimises them according to the specified fitness criteria. This operation is controlled by the user, who masters the parameters and fitness criteria. The ranking according to the automated fitness evaluation is a suggestion that the user can follow, but does not have to.

The main difference between the automated surface growth and the growth of surfaces in the evolutionary setup is the replacement of random element selection in the growth through the introduction of a genome. The genome is the genetic code for a particular global surface geometry. It can be revisited at any time and therefore allows the process of optimisation.

9.2. Artificial evolution - heuristic search

A rule which tends to give the right answer, but is not guaranteed to do so, is called a heuristic. Other optimisation techniques search the space of possible solutions by going through all the possible combinations step by step, but a heuristic method makes what can be called 'educated guesses'. Therefore, it is mostly faster than a linear search algorithm and almost always finds the right answer. Simulated evolution is an example of a heuristic technique.

As a course of optimisation, this technique became relevant to the industry with the development of computing power, enabling the artificial compression of time-based processes such as evolution. The process is applied to complex problems in which the ongoing search through the space of possible combinations is complemented by a learning process, where incorrect steps are taken out and correct steps are taken on to the next generation. The technique is applied where the number of possible combinations is so high that the regulators can be defined, but not the outcome. Writing about evolutionary techniques in his book 'The Pattern on the Stone', William Daniel Hillis says, "Its weaknesses as well as its strengths stem from evolution's inherent blindness to the 'Why' of a design. (…) evolution chooses variations blindly, without taking into account how the changes will affect the outcome" (Hillis 2001, c1998, p. 148).

The "inherent blindness" helps to diversify and, at the same time, filter the search. There is a good chance of finding novel solutions to a given and perhaps familiar problem.

Evolutionary computation (EC) covers a number of computational models which run mechanisms known from evolution. All are computer-based problem-solving systems. EC encompasses the following techniques: genetic algorithm (GA);

genetic programming (GP evolving program); evolutionary strategy (ES numerical optimisation); classifier systems (CS); evolutionary programming (EP evolving programs). GA can be adapted to a wide range of optimisation problems, an ES is used specifically for numerical optimisation, and GP, classifier systems and EP evolve programs. Therefore, GA was the obvious choice for this design implementation.

9.3. Genetic algorithm (GA)

The most common type of EA is the genetic algorithm (GA) (Mitchell 1998, c1996). In a standard GA structure, candidate solutions (phenotypes) are represented in genomes (genotypes), which are encoded as character or integer strings. Commonly, the first population of genomes is randomly initiated. The population is evaluated according to a fitness measure and the next generation is created by applying the operators: elitism, crossover and mutation. The breeding process of evaluation and reproduction is repeated, until either a specific result is achieved or, when reaching a specified number of generation runs, the process is ended.(Figure 136).

In short, what is needed for a GA is a genetic representation of the solution domain and a fitness function to evaluate the solutions in this domain. The main principle of simulated evolution is diversity, which is primarily ensured at the beginning of the process when the first population is created. The bigger the population, the bigger the search pool of potential solutions. The solutions might be 'seeded' in areas where optimal solutions are expected, meaning that the process is started with a genetic pool that is not completely random. This would typically be applied to solve detailed design problems. However, most of the time, the initiation of the genotype uses a pseudo-random function to allow for a broad search.

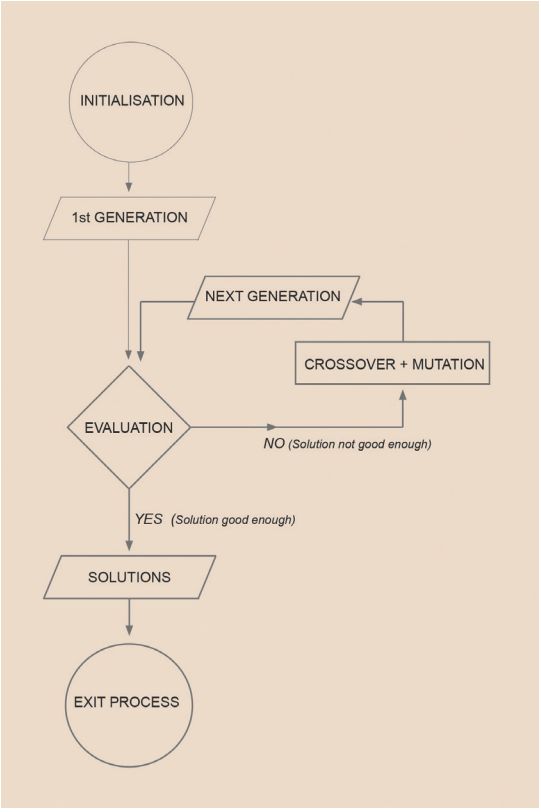


Figure 136
Flowchart of a basic genetic algorithm.

9.4. Fitness evaluation

The fitness criterion is the goal towards which the solutions are developed. It quantifies the optimality of a solution. The definition of fitness is the crucial core in the design of the evolutionary search method. There is no generic technique for evaluating fitness in architectural applications. Juan Romero and Penousal Machado (Romero, Machado 2008) suggest four different approaches: a) interactive evolutionary computation (IEC), b) learning user preferences, c) a rule-based system and d) co-evolving critics.

The most popular of these four is IEC, which bypasses the problem of defining a fitness function by having the user rank the members of each generation. This resolves the issue of defining fitness algorithmically, but introduces another problem, namely human fatigue. A user has a limited scope to objectively evaluate fitness in cases where the number of necessary considerations is high. Consequently, the population sizes and the numbers of generations must be kept low. This restricts the possibility of finding original designs (Lutton et al. 2003). The second approach is based on the idea of using a machine learning system, such as a neural network, for learning the user's preferences. This method does not work very well in practice, because there are too many parameters for the automated system to learn from a small training set. The third idea is to encode the user's knowledge about good designs as a set of rules. This approach works well for specific niche problems, but it is hard to make it work on more generic problems.

The approach used in the system discussed here is closest to the problem-specified, rule-based scheme. All methods imply that multiple objectives are considered. In this research project, it was decided that the fitness measures would be the overall area of the surface, created gaps, closeness to user-defined attractor points and the undulation of the surface (Figure 137). The first objective, the area, served as the test criterion to assess if the GA operators were working (page 217). The area gives clues about material usage and the attractors allow the description of a desired spatial arrangement. The surface is forced to branch with certain combinations of fitness criteria and parameter settings. These might be drastic orientation changes of attractors, together with a growth parameter setting where the element size relative to the distance of the attractor is big. The gap fitness allows one to find a solution which creates the minimum number of holes, while drawing close to or reaching the attractor points (Figure 138). The prior goal was to ensure that the full growth and optimisation cycle worked. The fitness objectives that were implemented could be extended to include additional fitness measures of spatial organisation or performance.

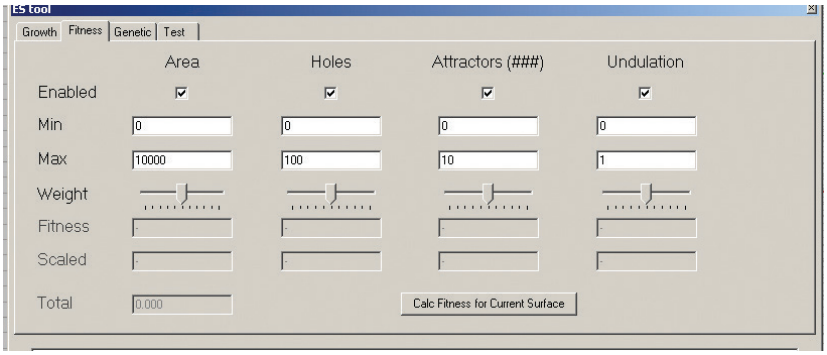
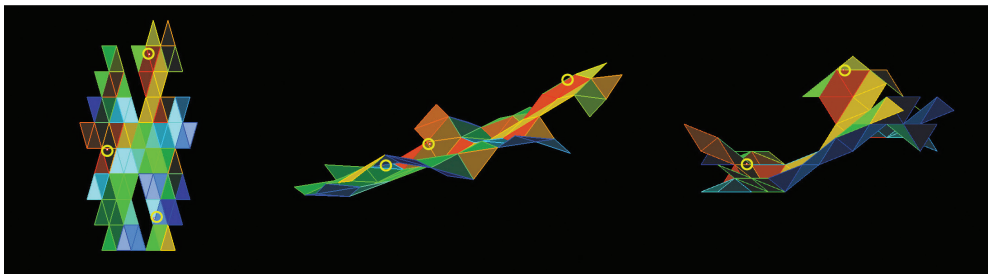


Figure 137
Graphical user interface of the fitness tab.

Figure 138
Attractor fitness. The screenshot shows an optimal solution in plan (left), isometric view (middle) and front view (right). The focus here is on the closeness to the three attractor points and the number of gaps created.



9.4.1. Multi-criteria optimisation

The design of an architectural surface requires a number of criteria to be considered, not just one. Aesthetics, spatial usability, structural performance, economic feasibility and ecological awareness are some of the factors that need to be taken into account at certain stages of the design process.

An interesting aspect of evolutionary search is that it allows for multiple fitness objectives to be employed at the same time. A challenge which arises when working with a number of fitness objectives is how to normalise and prioritise, namely how to communicate the different objectives, which might be incommensurable or conflicting. One example of conflicting criteria in the explored project on structural surfaces was the condition to achieve the most structurally stable roof surface possible, while at the same time creating the maximum space volume underneath. The criteria conflict, as the former means the structure would have to lie on the ground, but the latter demands that the roof would somehow hover in the air. Both objectives have to be communicated. Each influences the other, and subsequently, every decision taken opens up or constrains the pool of options to another decision to be made.

In GA development, it is common that every objective gets a 'weighting' applied to it, which can be described as a parametric rule. That is, each rule considers a particular feature, such as the maximum surface area of the envelope. The target value for this parameter, as well as its relative importance to other parameters, is defined by the user. The parameterization of the rules gives the tool greater flexibility and allows for a wider range of applications. Moreover, the traditional interactive evolutionary computation (IEC) features are retained, giving the user the possibility of micromanaging the selection. The various objective measures are then simply aggregated to form a single scalar fitness function. The danger of summing the diverse fitness values as a single function is that invalid solutions might gain a good fitness. This is commonly prevented by applying constraints to the variables. Here the product of the fitness is created instead of the sum of the fitness, for example 5 × 0 × 0 (discarded; negative sum = competitive behaviour) or 1 × 1 × 1 (some fitness; positive sum = cooperative behaviour).

Multiple solution outcome - design variation

An interesting characteristic of evolutionary search is that a population of solutions is always maintained and it is very likely that solutions within this population share the same fitness, despite their phenotypic differences. The aim is to find the Pareto

optimal set or front. A solution is a part of the Pareto optimal front when none of the objectives can be altered without worsening the others. Obviously, different trade-offs are possible.

In summary, the process supports the creation of a variety of solutions to a single design task. In applications where the Pareto front is too big, additional search techniques, or the so-called decision makers (DMs), are employed. In the prototype described, the DM is the user who either picks the preferred solution (subjective choice through soft decision factors) or allows different solutions to coexist. The ability to create alternative solutions to a single design task is a key motivation for this research. In this way, search becomes a creative means of exploration rather than a pure filter. It allows the study of parameter interdependencies and opens up the possibility of reconciling seemingly competing objectives, which can guide the user to unpredictable solutions.

9.5. Evolutionary parameters and operators

The growth parameters define the framework for the surface-generating process, while the evolutionary parameters and evolutionary operators create the outline for the optimisation procedure.

The evolutionary parameters are population size and the number of generation cycles. The search operators are the procedures which ensure that the process undergoes a permanent improvement of particular features. The search operators are: elites, tournament rate, crossover and mutation.

Population is the number of individuals in a generation.

Generations indicates the number of iterative cycles. In each cycle, a new population is created.

Tournament rate and **crossover** are linked parameters. Tournament rate specifies the number of randomly chosen individuals from the current population. Only the fittest of this subset are crossed with each other using the crossover operator. The lower the tournament rate, the more likely it is that a less fit individual is crossed, despite its unfitness. The result is a more differentiated solution pool. A high tournament rate might cause the process to converge to local optima. This is because it is more likely that the fittest enter the subset, and are therefore the ones to crossover and produce the offspring.

Mutation specifies the percentage of string bits which are mutated, meaning randomly changed. The percentage relates to a constant which is not influenced by any of the other parameters.

Elites indicate the number of fit individuals (the fittest, second fittest, third fittest and so on) which are copied straight into the next generation without any modifications. Elites are not touched by the mutation and crossover operations. With a high number of directly copied features of individuals, the process does not experience as much change in the genetic information as when the number of elites is low. Therefore, a high percentage of elites tend to force the process to experience a premature convergence to the local optimum.

In the arena of EA research and application, there is an ongoing discussion of how important mutation is for the success of evolutionary processing. The author suggests that it is a matter of the growth elements, thus the growth language which represents the seed. Its inherent rules allow or restrict diversity. In a course where the initial growth rules are as open to evolving structures as in the case of Genr8, the mutation operator is a fine addition, but not essential for the maintenance of multiplicity. Whereas for operations in which the initial seed is relatively limited, mutation is essential to ensure that a wide variety of structural configurations has been searched for optimal solutions and not been narrowed through the initial creation of genes for the first population. That is the case in the growth language that is developed and applied here. The parameter test section in this chapter shows the positive influence of the mutation on the performance.

9.6. The genome and its representation

The genome in the presented process is an integer string. Every string holds the history for a particular solution surface. Given the same genome and the same growth order, the same candidate solution is produced. Each integer, which refers to the gene in the genome, is the catalyst for placing one element in the surface; they are growth instructions like the chromosomes in human DNA. The length of the genome determines how big the surface can become, therefore it is linked with the user input for the surface size. If, for instance, the user inputs a surface size of 10 surface elements, the genome is 10 bits long (Figure 139).

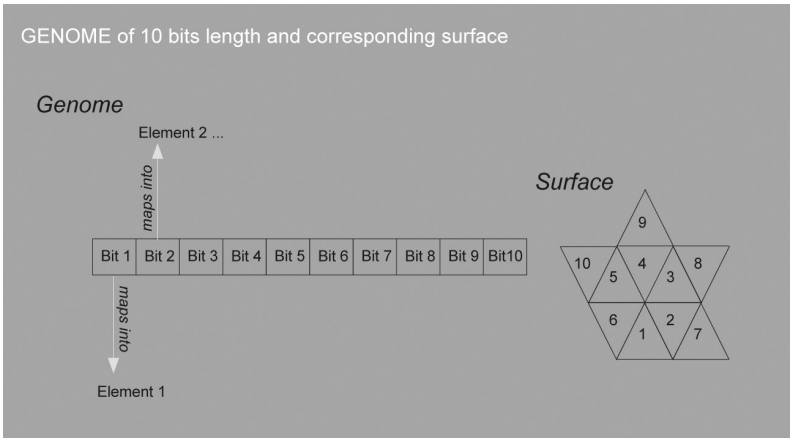


Figure 139
Genome. The illustration shows a 10-bit-long genome and the corresponding surface built from 10 elements.

9.7. Mapping the genotype into surfaces – two approaches

The decoding of the genome into its phenotypic surface representation, the so-called mapping, is an important design decision in the development of a GA. It is the feature control of the solutions. There can be various mapping techniques created. The method, however, must be consistent for the whole process, in order to ensure that a particular genome string carries the instruction for a particular surface individual, given a particular growth order.

In the developed prototype, two different mapping strategies are tested. The first uses the modulus function to decode the genome value into the next element that

fits. Here, this is referred to as 'MOD' mapping. In the second approach, named 'Best Fit', each gene value in the genome indicates a specific element to be placed into a position. If this particular element does not fit, then a local search is employed to try to find an element which most closely matches the directional features of the element that was indicated by the gene value. In both the mapping schemes, each gene in the genome gives the instruction to place one element in the surface. The feature which is controlled is the direction of the surface growth, whether it goes up or down, or stays at the same level. Every element has an inherent direction (Figure 140).

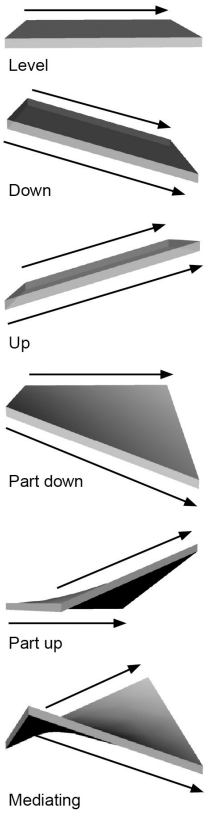


Figure 140
Element direction. Direction feature, illustrated here in the basic square tile family.

In Figure 141 and Figure 143, both mapping procedures are illustrated. In order to ease the understanding of the process, the growth model is derived from a simple orthogonal grid. The elements are outlined as surfaces and only a single z-step is allowed. Therefore, there are five different element types and ten corresponding rotations.

The basic element table is shown on the bottom left side of the first page in Figure 141. The elements with the same colour have exactly the same form but are rotated. The rotation of an element has to be considered as the growth process that has a spatial orientation. Consequently, the orientation of the growing edge is important and the rotated elements carry an independent index.

Type three and type four have the same element form but are flipped horizontally. Some physical definitions would demand a differentiation between the internal and the external material definition (composite). The two variations are therefore distinguished. The growth order for the illustration of both the MOD mapping and the Best Fit mapping was rotational and anticlockwise. Note that this is different from the program implementation, where the rhomboidal order was used.

9.8. MOD mapping - using the modulus function as a decision factor to choose possible elements

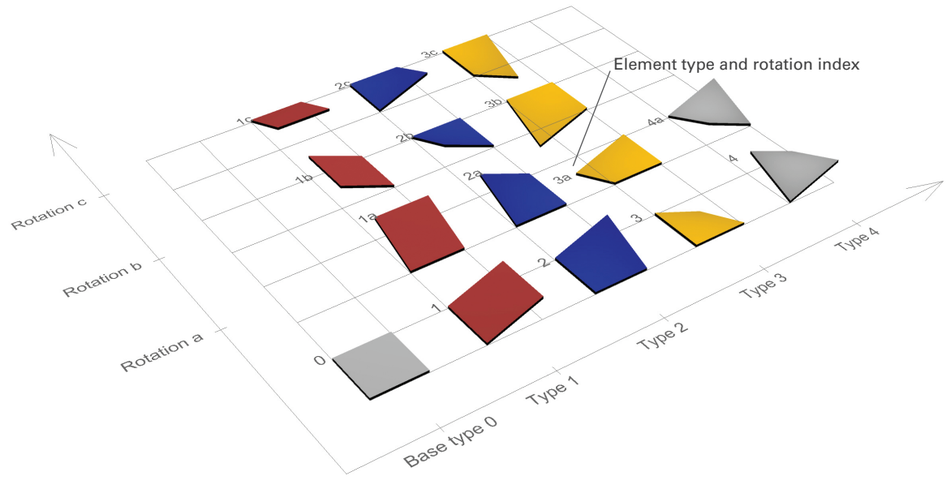
In the mapping which uses the modulus function, every gene value in the genome (every integer in the integer string) is a decision factor for which element from the choice of fitting elements is placed in the next position. The gene values do not indicate the specific element type.

The modulus function is used for calculating the remainder between the number of possible element forms and the integer in the gene. Hence, there is always a valid element which can be placed. The range of possible gene values is interrelated with the number of elements in the element table (Figure 141). The procedure creates a sensitive dependence on the initial conditions; a change in one gene value in the genome causes a change in the meaning of all of the following gene values.

Genome decoding using the ‘MOD’ function

In the mapping which uses the modulus (‘MOD’) function, every gene value in the genome (integer in the integer string) is a decision factor, for which element of all the possible elements is placed in the next position. The gene value does not indicate the specific element type directly. Instead the ‘MOD’ function is used to calculate the remainder between the number of possible element forms and the integer in the gene. This way there is always a valid element which can be placed. The range from which the respective gene value is randomly chosen should be big enough, in order to make sure that during the process the whole choice of elements gets a chance to be used. It must also be consistent throughout the process. The procedure creates a sensitive dependence on initial conditions.

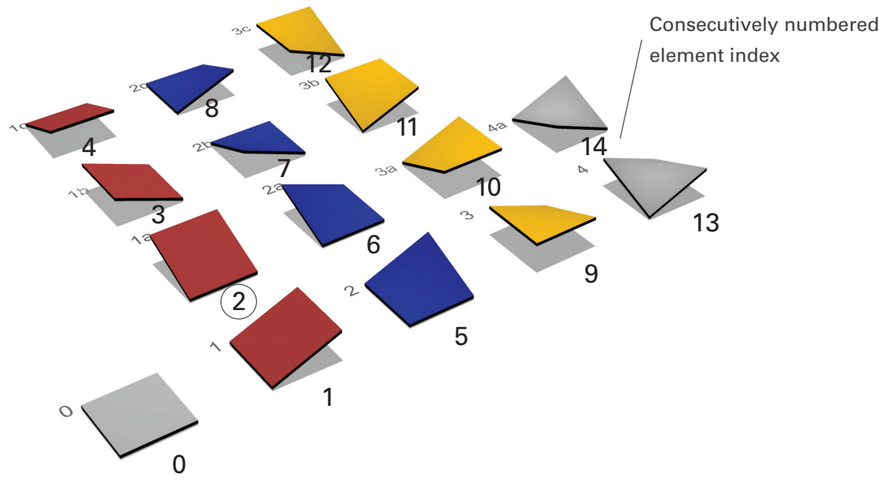
A change of one gene value in the genome causes a change in the meaning of all following gene values.



Basic element table in isometric view, showing five form types numbered from zero to four, and the corresponding rotations indexed as a, b and c. The elements with the same colour have exactly the same form but are rotated.

GROWTH STEP 1

Start of growth by placing the seed element



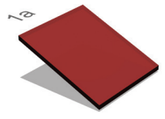
All elements could become the seed of the growth. therefore all elements enter the possible element list and 14 is inserted into the ‘MOD’ function.

2	1	7	5	7	2	1	0	6

Genome with the first gene value indicated and the corresponding ‘MOD’ function.

2 mod 15 = (2)

‘MOD’ function, dividing the count of possible elements by the gene value. The remainder is one. The blue element is placed next to the seed element.



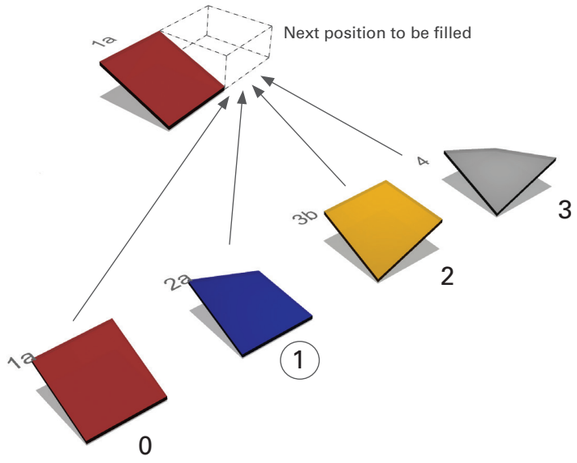
Selected seed element

In the first growth step the seed element is placed. As there is no restriction by any neighbouring tile, all fifteen elements of the basic element table are possible candidates. The ‘MOD’ function, therefore is 2 mod 15 = 2. The remainder is two. Counting through the list of possible elements, starting from nought, the red tile with the element table index three becomes the first tile of the growth.

Figure 141
Genome decoding using the ‘MOD’ function.

GROWTH STEP 2

Decoding of the second gene in the genome, placing the second surface element.



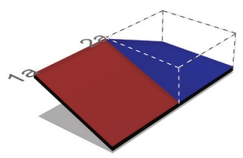
The four elements which fit into the empty position. According to their position in the base element table they are indexed for the possible element list.

2	1	7	5	7	2	1	0	6

The genome with the second gene indicated. The first has been mapped into the seed tile already.

1 mod 4 = (1)

'MOD' function, dividing the count of possible elements by the gene value. The remainder is one. The blue element (1) is placed next to the seed element.

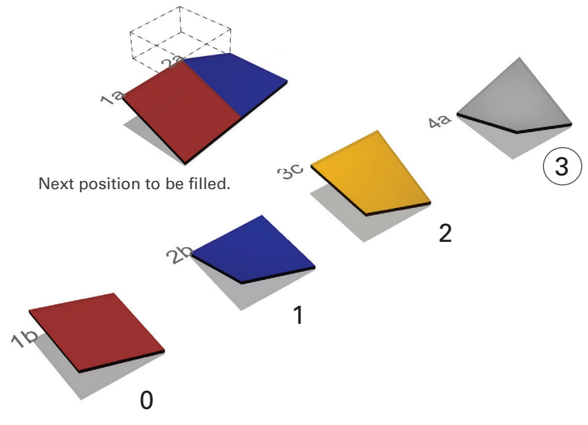


The blue element is placed, the surface grows.

In the second growth step an element is placed, filling the place to the right of the seed element. Four of the base elements can fit. The gene value is two; the 'MOD' function therefore is 1 mod 4 = 1. The remainder is one. The second possible tile is the element with the index 2a. It is placed to the right of the red tile.

GROWTH STEP 3

Decoding of the third gene in the genome, placing the third surface element.



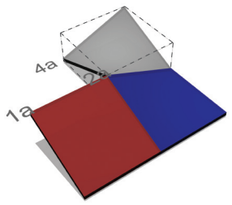
The four possible fitting elements.

2	1	7	5	7	2	1	0	6

The genome with the third gene indicated.

7 mod 4 = (3)

'MOD' function.

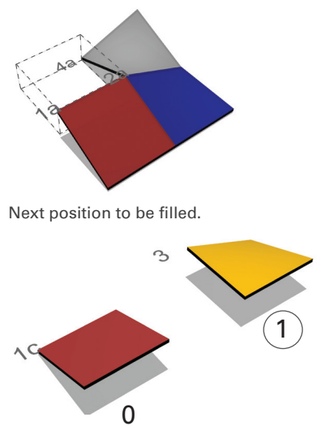


The grey element is placed, the surface grows.

In the next growth step the element connecting to the top edge of the blue tile is placed. Again, four of the base elements can fit. The gene value is four; the 'MOD' function is 7 mod 4 = 3. The remainder is three. The fourth possible tile of the list is picked and placed to the top of the blue element.

GROWTH STEP 4

Decoding of the fourth gene in the genome, placing the fourth surface element.

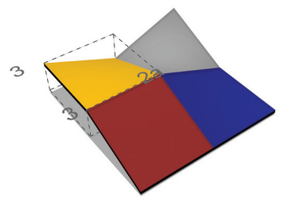


The two possible fitting elements.

2	1	7	5	7	2	1	0	6

The genome with the fourth gene indicated.

5 mod 2 = (1)



The yellow element is placed, the surface grows.

In the fourth growth step the element in the second hierarchy is placed, meaning the element which has to fit to two already defined edges. Only two of the base elements can fit. The gene value is six; the 'MOD' function is 5 mod 2 = 1. The remainder is one, therefore the second possible tile of the list is picked and placed between the grey and the red element.

9.9. Testing the 'MOD' mapping

As in cellular automata (von Neumann, W. Burks 1966), the procedure has two deciding factors: the state of the neighbours and the value of the gene. The chosen element is likely to be different if one of the two changes.

This form of mapping was employed in the surface generation tool ECSS (Jonas, Hemberg 2006), introduced earlier in this thesis. Here, the optimisation as well as the exploration was successful, which is why this approach was considered for this design implementation. However, the question was whether this mapping allowed for a correct crossover. The following experiment examined this concern. Two parent surfaces were crossed at the midpoint. The first three chosen elements sampled the parent that provided the first three genes. However, the second half of the surface was not identical with the equivalent part of the second parent, despite the fact that it was decoded from exactly the same integer sequence (Figure 142).

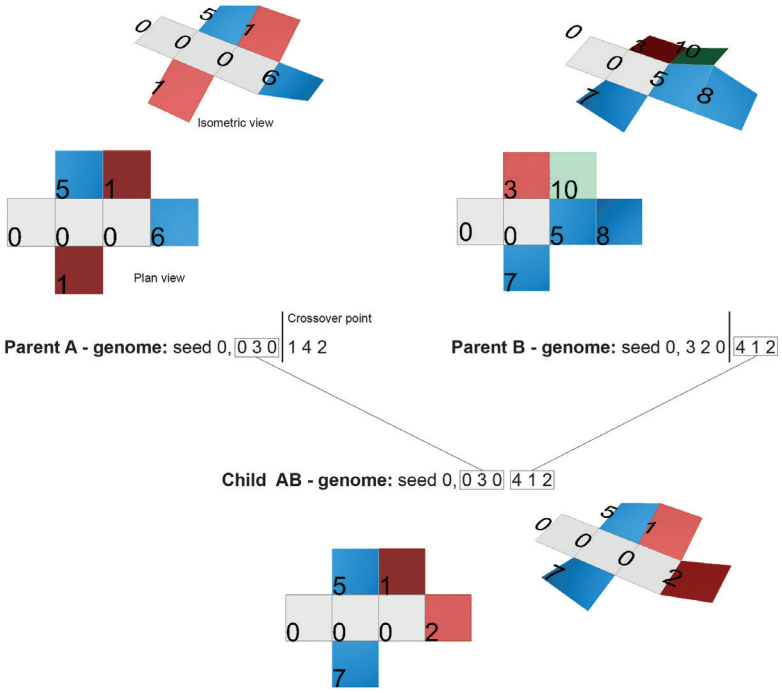


Figure 142
Crossover test. The test used the MOD mapping to decode the genetic information.

The child took on the features of the first parent and, depending on these, the integers in the second part of the genome from the second parent had a completely different effect than in the full sequence of the second parent's genotype. The sexual crossover became a sort of asexual reproduction, where a part of the successful candidate was taken over, while the second part was a new creation. One example of an asexual reproduction method can be found in evolutionary programming (Bentley 1999), where only mutation and no crossover is employed to create the next generation. However, for the tool development described here, which uses a genetic algorithm, the intention was to have an accurate crossover. Therefore an alternative mapping method was created and tested.

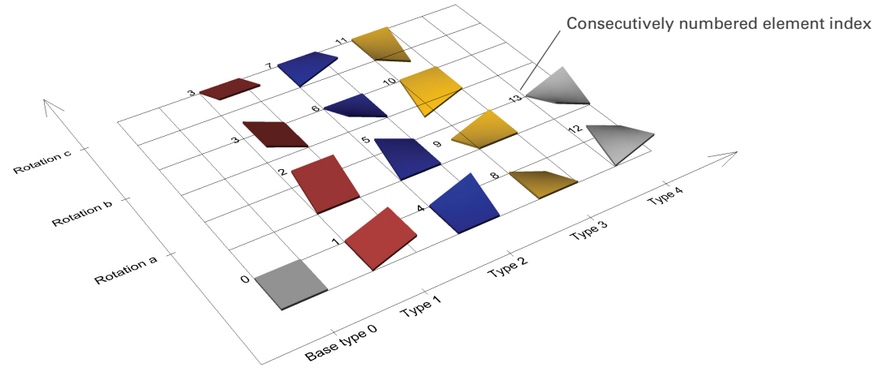
9.10. 'Best Fit' mapping – discrete tile mapping with local search

Other than in the MOD mapping, the gene value here indicates a specific, favoured element, therefore this element has to exist. To ensure that the element is present, the possible number range for the gene values is generated with respect to the user input, which determines how many different elements enter the process. In the likely case that the element, indicated by the gene value, does not fit into the next position, a search is conducted. This local search aids the finding of an element that fits and which most closely matches the directional feature of the element that was pointed out by the genome.

Genome decoding using the ‘Best fit’ function

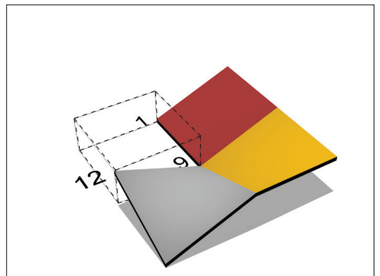
The diagram describes the so called ‘Best fit’ mapping process; how to decode the genome into a corresponding surface. The ‘Best fit’ method employs a local search operation to find the closest element match to the favoured tile.

Other than in the MOD mapping, the gene value here indicates a specific, favoured element; therefore this element has to exist. To ensure that the element is present, the possible number range for the gene values is generated in respect to the user input which determines how many different elements are entering the process. In the likely case that the element, indicated by the gene value, does not fit into the respective position, a search is conducted. This local search aids the finding of a fitting element which provides the closest match to the directional feature of the element that was pointed out by the gene value.

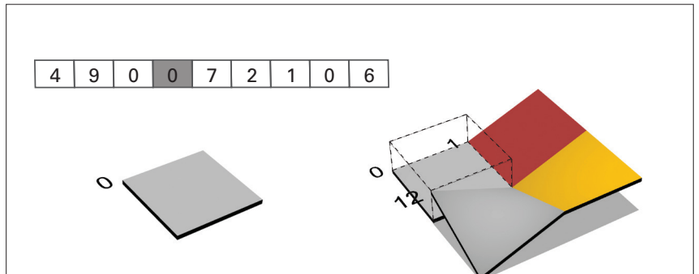


The tiles in the base element table are indicated linearly here. The numbers correspond directly to the gene value.

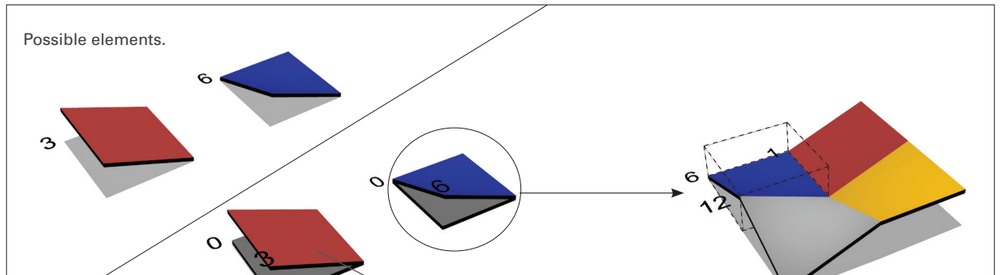
Placing one element using the ‘Best fit’ mapping



Current state of surface. The next element will be placed in the second hierarchy, meaning that the number of fitting elements will be fewer than in the first hierarchy, as there are already two edges defined. It will sit between the grey tile with the element index twelve and the red tile indexed with one.



The gene in the genome (top) for the space to be filled is zero (left). The tile with the element index zero is the desired form to be placed, but it does not match with the edges of the two constraining tiles (right).



Tiles three and six fit the new position (top). They are overlaid with tile zero (bottom). Element six has only two vectors which do not match, while element three has three dissimilar edges. Therefore, element six is chosen. Element six is placed into the surface formation.

Figure 143
Genome decoding using the ‘Best fit’ function.

9.11. Comparison between 'MOD' and 'Best Fit' mapping

Computationally, the MOD function is not as elegant as Best Fit mapping, as it operates a semi-accurate crossover. With respect to the actual outcome, however, it was interesting to compare both mapping procedures to find out which performed better. To do so, two evolutionary processes were carried out, one using the MOD mapping and one using the Best Fit mapping (Figure 144). The fitness objective of this test was to minimise the area of the surface. The fitness score of the fittest individual was printed at every tenth generation. The course ran for 250 generation cycles. The problem size was a surface of 64 elements in total, therefore the optimum fitness was 64. The resulting graph shows that even though neither of the two approaches reached the optimum during the test, they did perform an optimisation and both came close to the optimum. The red curve denotes the Best Fit mapping and shows a better execution than the blue curve, which indicates the MOD mapping. The Best Fit scheme was not only more elegant computationally but also performed better, and was therefore chosen as the default mapping procedure.

Comparison MOD - / Best Fit mapping

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		200	
Generations	0	250	10
Elites		4	
Tournament size		2	
Mutation rate [%]		3	
Iteration runs		15	

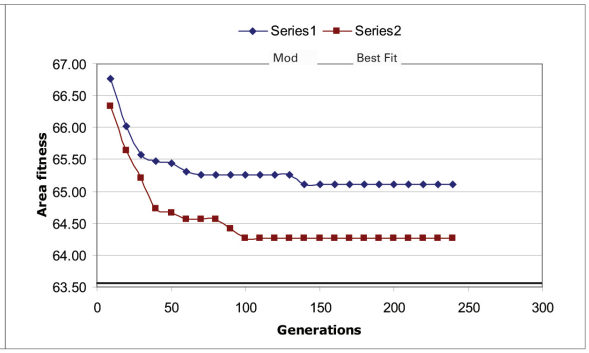


Figure 144
Comparison between the MOD and the Best Fit mapping.

9.12. Testing the genetic algorithm and its operators

9.12.1. Introduction

This section describes the performance test of the GA. The reason for conducting this test was twofold. Firstly, it was necessary to ensure that the optimisation process was working. Secondly, the dependency between the individual operators could be investigated. The test also allowed suitable default parameters to be defined for the GUI.

The parameters were changed sequentially. Each setting was kept for a number of iterations to verify the results and the outcome was captured in a graph.

In order to monitor the performance of the parameters and the overall optimisation process, a single fitness was implemented, against which the test results could be easily measured easily. The single test fitness was the area of the surface and the fittest individual was the one with the smallest area. The surface area increased with the degree of articulation, making the optimum individual a planar surface. When the surface size or the element height was amplified, the differences between the areas taken up by individuals increased. In a real project application, the area of a surface is an indicator of material usage and thereby an economic factor. To avoid the creation

of solely planar solutions, it would need to be complemented by additional spatial measures. When testing the GA, however, the area as fitness on its own was a good measure of the optimisation performance.

9.12.2. Setup of the GA parameter test

The first parameter investigated was the number of generations. It was important that the process ran for a sufficient number of generations to ensure that the global optima were found and the evolution did not come to a premature halt. At the same time, to avoid inefficient computation, the aim was to avoid running an unnecessary number of cycles. Once a suitable number of generations was found, the next parameter was tested, which was the size of the population. The population, thus the number of individuals in every generation, was a crucial factor, as every individual brought in a new genotype. Thereby, with an increase in the size of the population, the variation of genotypes entering the process increased. During the generation runs, this initial gene pool of the first population was altered through the genetic operators. There were no new genotypes entering the process, which is why it was desirable to start with a large number of individuals.

After a 'good' generation and population number were found, a check was performed on the remaining operators, which were harder to predict. The generation and elites parameters were revisited at the end of the test (page 225), after the default setting for each of the other parameters was defined.

The graphs mostly display multiple iterations overlaying each other, in order to verify the performance trend. In some instances, the average is shown as well.

The graphs presented here are advanced test results, which were obtained after an elaborate development process in which the software was debugged and the test setup improved. One of the major changes to the parameter encoding was the change from an explicit number input of the elites and tournament size to a percentage input. From a user interaction point of view, this change appeared to be rather unintuitive. However, it seemed essential for the purpose of monitoring the performance, especially that of the population parameter, because it was the only way that the operator could assure that the operators (elites, tournament size and mutation) did not influence the outcome.

The tests shown here are a third of the examinations undertaken, although they include the changes from the actual numbers for the elites and the tournament input to their percentages, plus the allowance to input decimal increment steps. The first test, which is the examination of the generation parameter, included the immediate translation of what was found to be a good setting from the previous set-up. This report goes through all the other parameters again, to document the performance trend of each constraint. It is purposely not presented in a linear fashion; this shows how an understanding is gradually established of the individual parameters' performances and their dependencies to each other.

They might not have been set to the best value at the point of the population test, but their influence on the process stayed the same, because their actual number increased in proportion to the population size.

9.12.3. Test interface

The test interface allowed the definition of the start and the end number for a particular parameter, as well as the stepping. This allowed stepping over a few values to accelerate the process.

In the example, the test (Figure 145) started with 10 generations; each generation had a population of 50 surface individuals and ran until the 500th generation had been reached. The fitness score of the fittest individual was printed at every 10th generation step, therefore at 10, 20, 30, 40 and so on. When the 500th generation was reached, the procedure started from the beginning. The whole procedure was repeated for 10 iterations.

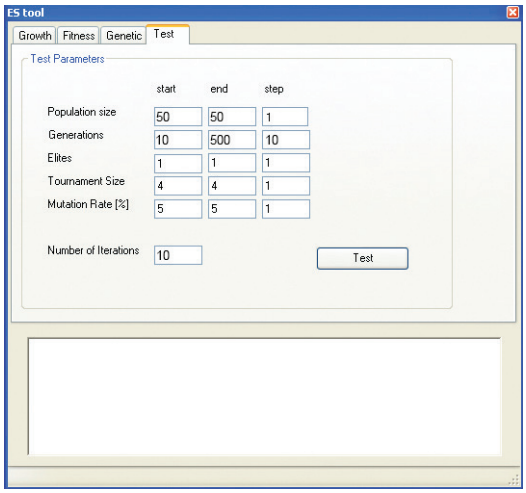


Figure 145
Graphical user interface for the GA test.

9.12.4. Premature convergence and approximation of the optimum

One of the fundamental ideas of using a heuristic search operation is to avoid converging at a local optimum. If one imagines a landscape full of small and large mountains, the aim is to find the highest peak. The smaller mountains indicate the local optima, while the taller mountains indicate better or even the best achievable solutions. There might be a number of similar or equally high mountains. Multiple solution outcomes with the same or similar fitness score are a characteristic of an evolutionary search when multiple fitness criteria are employed.

Mutation and tournament selection are operators used for diversifying the search, whereas elites help to maintain the best individuals in every generation. In some of the following tests, e.g. the revisited generation test in Figure 158, it was impossible to tell if two of the curves had converged to the local optima or if they would progress, given a wider window of generation runs. Theoretically, one would assume that the optimum must be reached at one point; practically, however, the change necessary to overcome a valley in the fitness landscape might be bigger than that supported by the operator setting. The mutation parameter, in particular, had a great influence on whether a valley could be overcome. The optimum should be reached eventually with the single fitness, given the correct setting of the mutation. The aim of the test was to balance both the seemingly conflicting procedures: widening out and congregating the solution pool.

9.12.5. Generations

The parameter population and the operators (elites, tournament selection and mutation rate) were fixed. The performance of the process under a linearly changing number of generation runs is observed. The surface size is set to 8 × 8 tiles, so the optimum was a minimum area of 64 square units.

The graph in Figure 146 shows three iterations of generation runs. The area of the fittest individual at every generation step was printed. A clear descending curve was exhibited, showing the process improved steadily. At the 55th generation, the optimum was reached in all three generation cycles. The process had converged. It can be assumed that with this setup, 60 generations were sufficient to reach the optimum results.

Generation test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations	0	100	1
Elites [%]		0.5	
Tournament size [%]		0.3	
Mutation rate [%]		5	
Iteration runs		3	

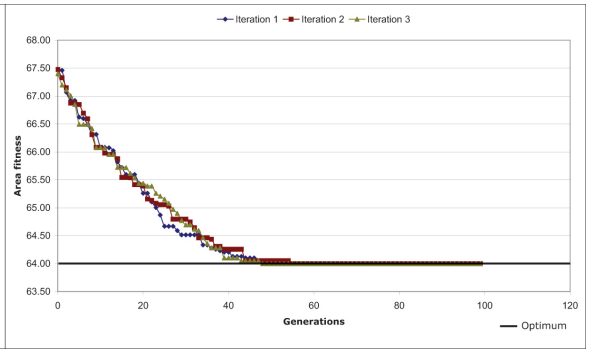


Figure 146
Generation test graph.

9.12.6. Population size

A general rule is that the higher the number of individuals in a generation, the greater is the pool of different initial genomes. In genetic algorithms, the genomes are initiated in the first generation. They define the basic heritage for the whole process, as the genomes of the subsequent generations consist only of variations of the initial genomes, not newly initiated ones. The variations are created through the application of the evolutionary operators. Nevertheless, there is a point of convergence and the process should not create more than the necessary number of individuals, as this is computationally expensive.

When investigating the population size parameter, the generations and the operators (elites, tournament selection and mutation rate) were fixed. The performance of the process under a linearly changing size of population from 10 to 1050 was monitored. The surface size was set to 8 × 8 tiles, so the optimum was an area of 64 square units. Every point on a curve indicated the best fitness reached at the end of a 100 generation cycle with a specific population size.

The next point then showed the best fitness reached at the end of 100 generations with a population that had 10 more individuals than in the previous cycle. This caused the curve to fluctuate locally. The process was continued from a size of 10 to a size of 1050 individuals in a generation, and the whole process was repeated three times.

At first, the chart in Figure 147 presents a slow decline of the three curves, then a dramatic fall when the population size reaches 500. The reason for the abrupt fall was that with a population of 500, the number of elites jumped from 2 to 3 individuals which were copied into the next generation and the tournament rate flipped from

1 to 2 selected to perform the crossover. It was therefore not the increase in the population that influenced the performance most at the population mark of 500, but the tournament rate operator. When looking at the tournament rate test (page 223), it is evident that the change from 1 to 2 actually selected individuals made an important difference and improved the overall performance. From there on, the curves decline slowly until they hit the optimum line more frequently between 980 and 1020.

Population test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size	10	1050	10
Generations		100	
Elites [%]		0.5	
Tournament size [%]		0.3	
Mutation rate [%]		5	
Iteration runs		3	

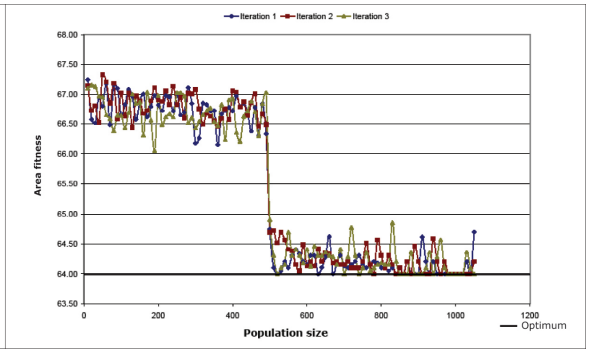


Figure 147
Population test graph. The graph displays a steep jump, which can be explained by the change in tournament selection. At the population mark of 500, the tournament selection changed from one to two.

9.12.7. Elites

The elites parameter, as with the tournament size parameter, was set to a percentage relating to the number of individuals in the population. This meant that with an increase in the population size, the number of elites increased linearly. Each point in the test diagram (Figure 148) shows the fitness of the fittest individual in the last generation. Therefore, each point marks an end point of every generation cycle, which is why the curves fluctuate locally.

The first test run covered the whole range from 0% to 100%, meaning from 0 to 1020 individuals of the population were copied as elites into the next generation.

Figure 148 shows an obvious trend: the optimum is reached relatively often between 0% and 50%. The curve bends slightly upwards between 50% and 75%, and then flies up, away from the optimum. When too many 'best' solutions were copied into the next generation, then the variation had fewer chances of bringing in new suggestions. This meant that the process could converge more easily to a local optimum, rather than a global one. Therefore, continuously increasing the number of elites was not beneficial; the right ratio has to be found.

Elites test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		100	
Elites [%]	0.04	100	1
Tournament size [%]		0.3	
Mutation rate [%]		5	
Iteration runs		5	

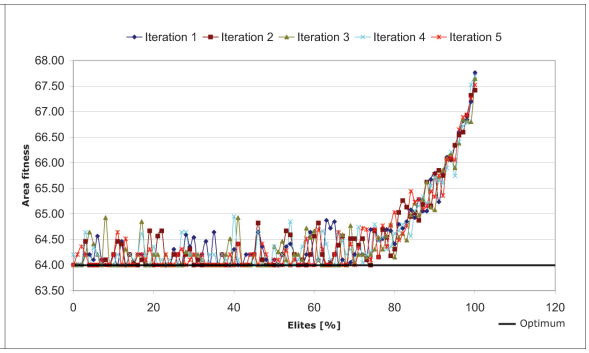
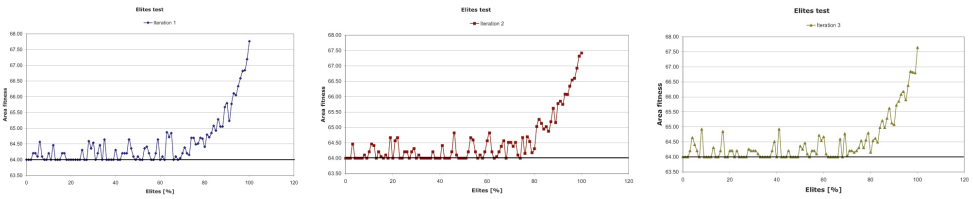


Figure 148
Elites test graph.

Figure 149 displays three of the five iterations of this initial test, indicating there might be an area in the range between 0% and 50% in which the curves come closer to the optimum line. In the first of the three graphs, the curve is nearer to the optimal area fitness between 18% and 24%; in the second and third graphs, there seems to be a better range between roughly 19% and 40%.

Figure 149
Three iterations. The graphs show three iterations of the first run, testing an elite parameter setting between 0% and 100%, thus between 0 and 1020 elites.



The average curve created from all five iterations (Figure 150) was low at the beginning, from 0% to 2.4%, with a few more even lower points between 18% and 38%.

Elites test average

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		100	
Elites [%]	0.04	100	1
Tournament size [%]		0.3	
Mutation rate [%]		5	
Iteration runs		5	

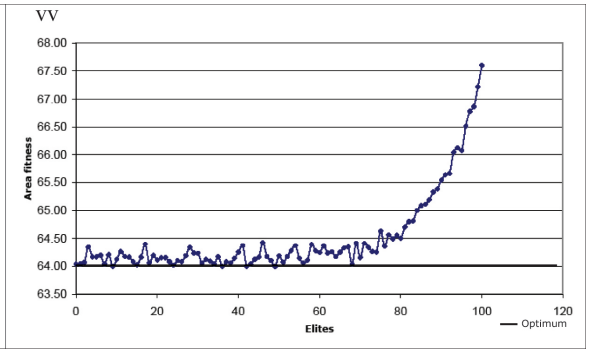


Figure 150
Elites test average curve.

It was not obvious why the performance was good again in the range between 18% and 38%.

In order to see if it was indeed a range in which good solutions could be found, the next test considered the range from 18% to 41% elites; 183 to 408 individuals of the population, respectively, were copied as elites into the next generation. Only a single iteration was performed. When first observing the results, the curve seemed to jump a lot. On closer inspection, 51.74% of the cycles hit the optimum (Figure 151).

Elites test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		100	
Elites [%]	18	41	0.05
Tournament size [%]		1	
Mutation rate [%]		5	
Iteration runs		1	

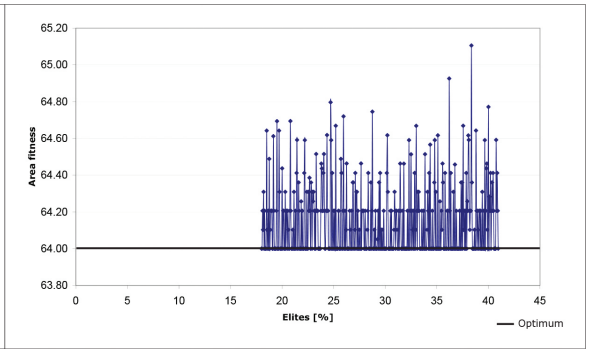


Figure 151
Close-up elites test. In the range between 18% and 41%, 51.74% of the cycles reached the optimum area fitness.

In an elites test, which it was assumed would be the final one, mutation was frozen to see if it affected or clarified the performance trend of the elite changes.

The graph (Figure 152) seems similar to the first elites test shown in Figure 148. Both have the same trend, the only difference being that the curve shifted upwards, meaning that the optimum was reached less often.

Elites test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		100	
Elites [%]	0.04	100	1
Tournament size [%]		1	
Mutation rate [%]		0	
Iteration runs		3	

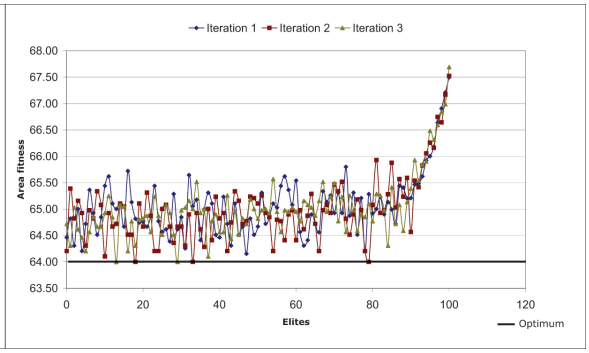


Figure 152
Elites test without mutation. The runs were repeated three times with the same setting.

9.12.8. Tournament rate

The tournament parameter was set to a percentage rate which relates to the number of individuals in the population.

The chart in Figure 153 shows the optimum was reached between 0% and 1.03%. This refers to a range of 0 to 11 selected individuals. Between 1.03% and 35.03%, the curve slowly rises again to finally alternate within a specific range. This means that the trend was neither rising nor falling from 35.03% onwards. Good tournament rate values seemed to lie between 0.03% and 1.03%. This assumption needed confirmation through the examination of a narrower range.

Tournament test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		100	
Elites [%]		0.3	
Tournament size [%]	0.03	100	1
Mutation rate [%]		5	
Iteration runs		1	

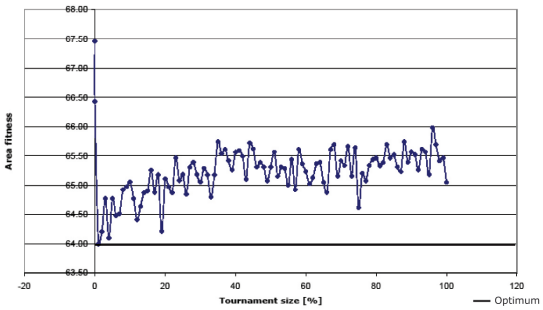


Figure 153
First tournament test graph.

Figure 154 displays the last tournament rate test in this sequence. The test looked at the range of 0% to 2.03%. The test aimed to clarify where the best results were. There were only two iterations explored, and the curves hit the optimum between 0.18% and 0.28%. The first iteration (blue) continued on the optimum line until 0.43. When converting the percentage rate into the actual number of individuals, the setting 0.18% indicated two individuals and 0.28% indicated three. Considering this setting, any value picked between 0.18% and 0.3% had to be reliable.

Tournament test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		100	
Elites [%]		0.3	
Tournament size [%]	0.03	2.08	0.05
Mutation rate [%]		5	
Iteration runs		2	

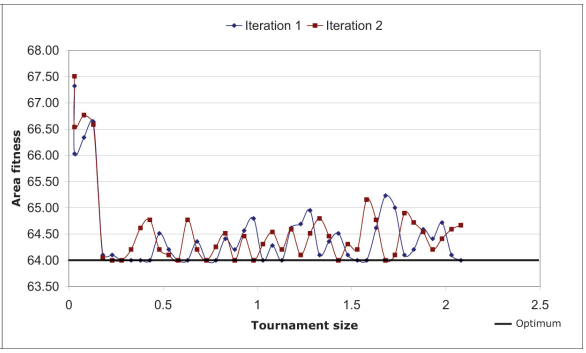


Figure 154
Second tournament test graph.

9.12.9. Mutation rate

Mutation specifies the percentage of string bits which are mutated, meaning randomly changed. Note that in this code, design mutation is not applied to elites. The percentage relates to a constant which is not influenced by the user input.

Figure 155 displays three iterations. All three exhibited the same steep, downward trend towards the optimum, between 0% and approximately 8%, followed by a similarly fast increase, which seemed to slow down at around 20%. The optimal value lay between 4% and 12%.

Mutation rate test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		100	
Elites [%]		28	
Tournament size [%]		0.3	
Mutation rate [%]	0	100	1
Iteration runs		3	

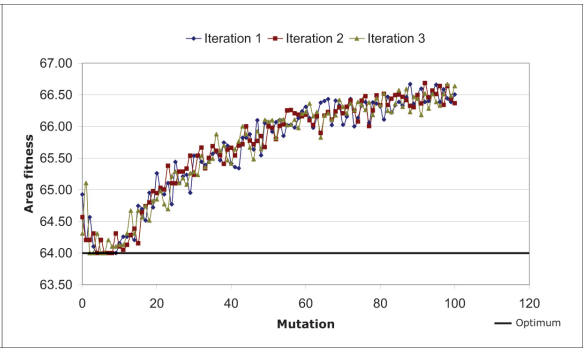


Figure 155
First mutation test graph.

A second test was conducted to look at the local range between 0% and 20%, scoping the area of the most extreme change. Figure 156 illustrates the three iterations of this test; the average is plotted in Figure 157. The first iteration created optimal solutions between 4% and 11%. However, iterations 2 and 3 produced both poor and optimal solutions within this same band: the area fitness was above the optimum mutation rate between 6.64% and 8.64%, only to reach the optimum again at 9.64%. As a result, where the lowest point of the valley would have been expected, the average curve shows a bump instead. Three iterations were not enough to make any quantitative assumptions, but following the average curve it seemed reasonable to choose a 7% mutation rate as the one setting that was likely to achieve the optimum area fitness.

Mutation rate test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		100	
Elites [%]		28	
Tournament size [%]		0.3	
Mutation rate [%]	0	20	1
Iteration runs		3	

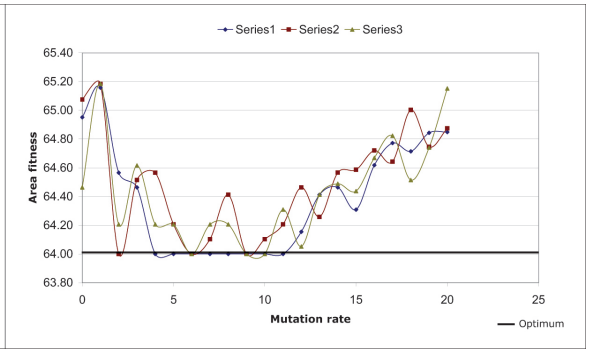


Figure 156
Close-up mutation test. Mutation test on the local range between 0% and 20%.

Mutation rate test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		100	
Elites [%]		28	
Tournament size [%]		0.3	
Mutation rate [%]	0	20	1
Iteration runs		1	

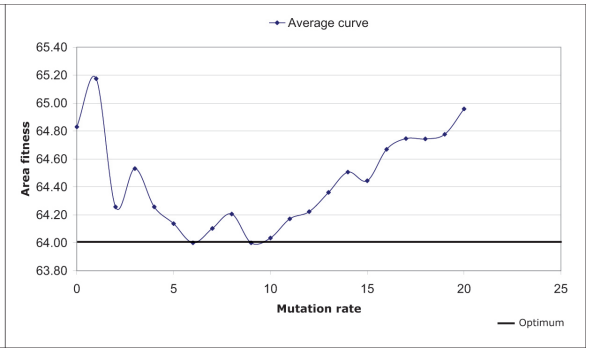


Figure 157
Average graph of the second mutation test.

9.12.10. Revisiting generation test and altering the elites input

After all the parameters were tested, the generation test was revisited; setting all other parameters to the setting which was found to be the best. In Figure 158, the elites value of 28% was taken from the range which was investigated in more detail during the elites tests.

Six of the eight iterations reached the optimum. All seemed to converge at around 100 generations, except iterations 2 and 7. From this test, it was not possible to tell whether they would converge at the optimum at one point or reach a local optimum.

Generation test revisited A

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations	0	500	1
Elites [%]		28	
Tournament size [%]		0.3	
Mutation rate [%]		7	
Iteration runs		8	

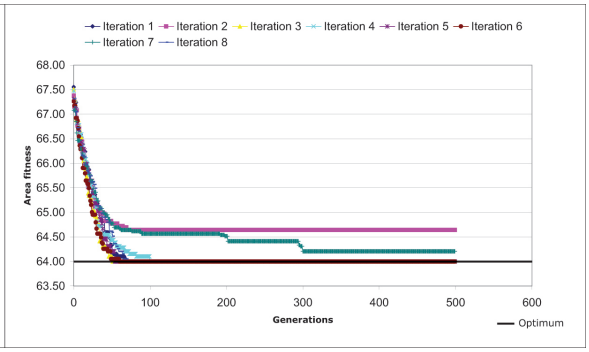


Figure 158
First revisited generation test graph.

The elites parameter was the most uncertain one in the test sequence. The previous generation test was repeated with a smaller elites value (10), to see if it improved the outcome further (Figure 159).

The resulting graph is flatter than the previous one: 7 out of 8 generations converged to the optimum. Iteration 1 reached the optimum a little later than the others, after 127 generations. In iteration 7, the results converged to a local optimum, 0.2 units above the global optimum.

Tournament test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		100	
Elites [%]		0.3	
Tournament size [%]	0.03	100	1
Mutation rate [%]		5	
Iteration runs		1	

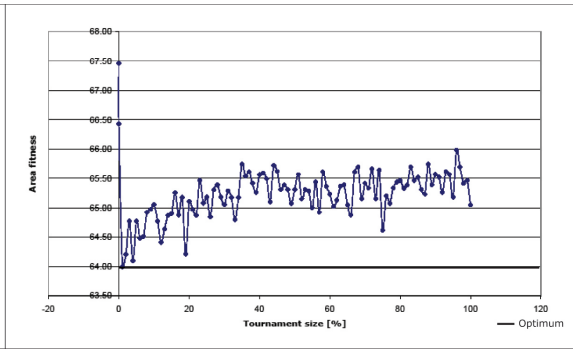


Figure 159
Second revisited generation test graph.

The test suggested that a lower elite value might be better. To verify this, two more elites tests were conducted. In the first test, a general run was repeated, testing the percentage values from 0% to 100% elites. The generation run was increased from 100 to 200 instead, to make sure that the process was given enough time; the increment step is 1.

The scale in which the curves in Figure 160 fluctuate is generally closer to the optimum than in Figure 148. The only difference in the settings was an increase in the number of generation runs. This makes sense when observing the number of generations that the process needed to converge in the generation tests, as shown in Figure 158 and Figure 159. Again, there appears to be an area between 20% and 40% in which the results were better; the points are visually closer to the optimum.

Elites test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		200	
Elites [%]	0	100	1
Tournament size [%]		0.3	
Mutation rate [%]		7	
Iteration runs		3	

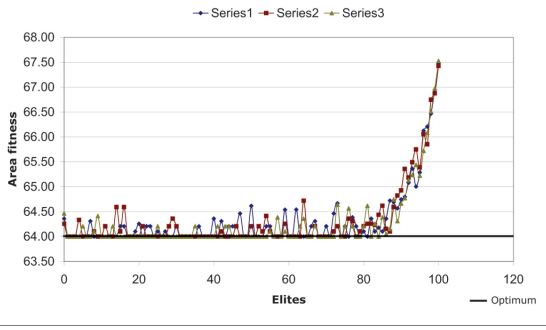


Figure 160
Elites test.

The second test looked at small percentage values from 0% to 4%, with a smaller incremental step of 0.05 (Figure 161). 56.8% of iterations 1 and 3 reached the optimum line; 58% in iteration 2. This was better than in the test shown in Figure 151, which looked at the local range between 18% and 41%, that exhibited a 51.74% success.

Between 0.05% and 1%, there was no run which deviated from the optimum by more than 0.38 units. Therefore, it was assumed that the most suitable elites setting lay in this range.

Elites test

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations		200	
Elites [%]	0	4	0.05
Tournament size [%]		0.3	
Mutation rate [%]		7	
Iteration runs		3	

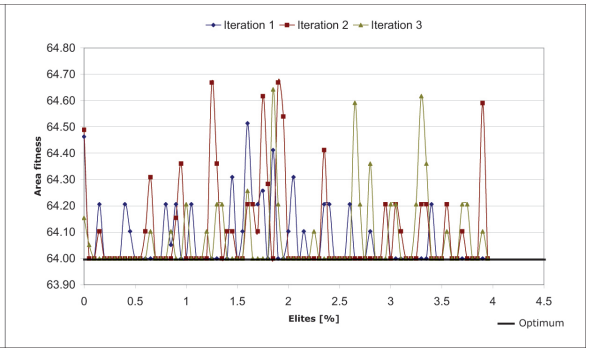


Figure 161
Second revisited elites test graph.

The generation test was repeated with an elites percentage value of 0.4% (Figure 162). The result was similar to the examination run with 10%, where 7 out of 8 iterations reached the optimum: the results of one iteration seemed to converge to a local optimum 0.2 units above the global optimum. After the whole sequence of elites tests, it was assumed that a small percentage value between 0.05% and 1% would reliably produce results that were optimal or close to the optimum.

To verify why the global test graph shows a flattening of the curve between 18% and 41%, and why (against expectations) the test with 10% produced the same result as the one with 0.4%, a quantitative test had to be conducted with many more iteration runs. However, the small values served a good purpose. This was confirmed in all the tests and is in accordance with what has been said about the elites in the parameter description: a high elites value is likely to cause the process to converge too early.

Generation test revisited D

	From	To	Steps
Surface size [Tiles]		8*8	
Population size		1020	
Generations	0	500	1
Elites [%]		0.4	
Tournament size [%]		0.3	
Mutation rate [%]		7	
Iteration runs		8	

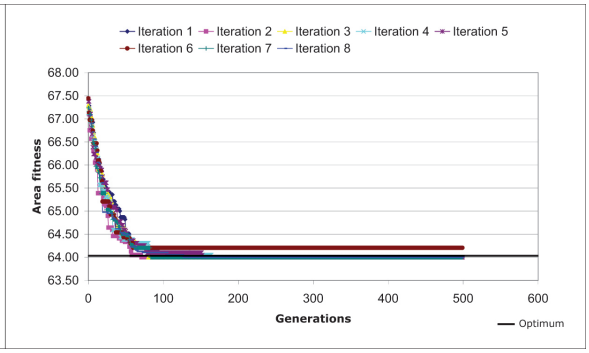


Figure 162
Third revisited generation test graph.

9.12.11. Influence of the problem size on the definition of a good parameter setting

All tests were conducted using an 8 × 8 unit surface. The next step was to carry out another test with a bigger problem size. The surface size was increased to contain 18 × 18 elements. The best area fitness was therefore 324 square units. It was assumed that there is some sort of linear relationship between the problem size and the necessary size of the population, or number of generation runs. A test had to be conducted using a proportional increase of the setting which was found to be good for a surface size

of 8 × 8 elements. 5164 individuals and 1012 generations were executed in the next test, which was proportional to the setting used for the 8 × 8 element-sized problem. Elites number and tournament size were automatically increased with the population size, as they were a percentage of it. The resulting graph (Figure 163) shows that the overall process improved, but the curves stay far from the optimum. A percentage increase in the setting that was found suitable for a smaller surface did not produce reliably good results for the bigger surface. That was the case, even after linearly increasing both population size and generations. From just this one test, it was not possible to establish whether the relationship is linear. A further question was whether the relationship between the individual operators' settings could stay proportional or needed alteration as well.

Bigger problem test

	From	To	Steps
Surface size [Tiles]		18*18	
Population size		5164	
Generations	0	1000	50
Elites [%]		0.4	
Tournament size [%]		0.3	
Mutation rate [%]		7	
Iteration runs		3	

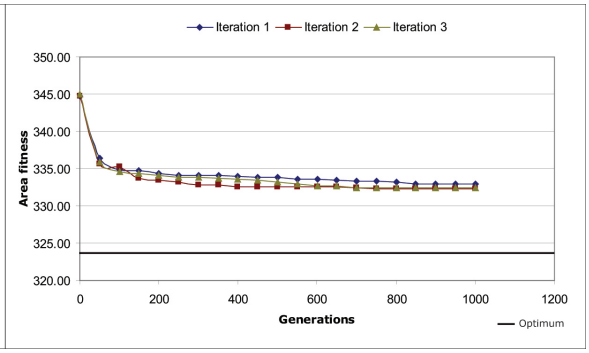


Figure 163
Graph of the fitness performance with a bigger problem size.

9.12.12. Conclusion from the GA test and further suggestions

The GA test aimed to answer two questions: first whether the process worked, i.e. if it optimised the solution pool, and second, how an understanding of the dependencies between the individual operators could be gained.

The answer to the first question is positive: the graphs which display the generation and population tests indicate a clear improvement of the solution pool over the cycle of generation runs. The process is correctly designed and implemented; it can, in principle, produce optimal solutions. The chance of success depends on the settings of the evolutionary operators.

This leads to the second question. The test sequence shows a good setting for a particular problem size: a surface of 8 × 8 elements in relation to a single fitness, the criteria to find the minimum possible area. The answer to the second question is given in the next paragraph. It comprises the findings of the individual parameter tests and the example of a good parameter setting.

The generation tests showed that a good setting for the parameter depends on the size of the problem and the size of the population. The bigger the surface, the longer the process needed to run in order to converge. It could be observed that the higher the number of individuals in a generation, thus the bigger the population, the fewer generations that were needed. There is an immediate dependency between the two operators. Primarily, optimal solutions are created, with a setting of 1020 individuals and around 200 generations for a surface size of 64 elements. In the tests, the optimum was already reached at around 100 generations. However, we

know that the longer the generation runs, the better. We also know that increasing the number of generations increases the computational cost. It therefore can be assumed that running two hundred generations is a good compromise. The question remains whether there is a linear correlation between the surface size, the size of the population and the number of necessary generations. So far, the test only verified that there is a dependency between the three constraints.

The population size indicated the range of unique genomes which enter the evolutionary procedure. It can be said that the higher the number of individuals in a generation, the greater the pool of different initial genomes. The succeeding generations simply consist of alterations of the initial genomes. The variations are created through the application of the evolutionary operators. Consequently, the size of the population should be high. The test proves that the size of the population has a more immediate effect on the performance of the code than the number of generation runs. It is computationally less expensive to maintain a large population than to increase the number of generation runs. As stated for the generation parameter, for a surface size setting of 64 elements and 200 generations, a population size of 1020 individuals is a reliable default setting.

The elites test demonstrated the consensus that too many elites produce a premature convergence at the local optima. The test result suggests that an elites setting smaller than 1% of a population of 1020 individuals is good. The recommended input of 0.4% related to 4 elites members of 1020 individuals of the population, which were copied into the next generation without any modification.

The tournament selection test indicated that a surface size setting of 64 elements, between 0.18% and 0.3% (meaning 2 and 3 individuals of the population which has 1020 members), should be randomly selected for a crossover. This ensured the facilitating performance of this parameter.

It could be advised to choose a mutation rate between 2% and 9%. A surprising result was the elites test in which the mutation was set to 0. The graphs in Figure 148 and Figure 152 prove how important the influence of the mutation operator is for the process. It does not disturb the overall trend but improves the chance of finding better solutions. This observation is interesting and crucial, as GAs are often criticised for being 'too random'. In this case, the pseudo-randomness prevented the process from stopping too early. In summary, the settings which are likely to produce an optimal surface are the following:

Example problem size:	64 elements
Population size:	1020 individuals
Number of generations:	200 runs
Elites:	0.4% of population
Tournament rate:	0.3% of population
Mutation rate:	7%

There was a tolerance in the settings, and the selected values were picked around the middle of a possible range to ensure that the process was likely to be successful while staying computationally efficient. When these settings were used in the actual

run mode of the program, the results were convincing, as the sequence in Figure 164 shows. Each screenshot displays a surface with the highest fitness score at a particular iteration. The generation runs which were completed increase from left to right. The screenshot on the far right shows the solution after the generation runs were completed, with the settings working well for a surface size of 8 × 8 elements. It therefore shows a planar solution; the global optimum was reached. The test sequence ensured that the code was working. To provide the suggested settings for the operators, however, a number of further tests are suggested. This is so that the ratio between the settings and the problem size, as well as the dependency on the fitness objectives, are fully understood.

It should be underlined that the tests in this chapter consider an artificial and not a real problem. The objective of the minimum area is only sensible for a real project application, if it is looked at together with another fitness criterion which controls the spatial articulation. The minimum area can stand in for the control of material usage, which is interesting when a specific spatial distribution is desired that can be described through attractor points. The fitness objectives that are included in this design implementation are described in paragraph 9.4 of this chapter.

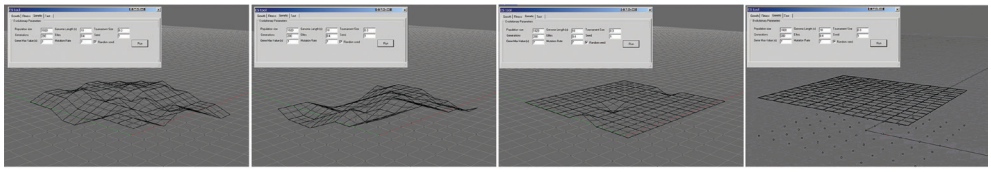


Figure 164
Minimum area optimisation. The image sequence shows a generation run which reaches the optimum (far right), after 200 iterations with a population size of 1020 individuals were completed. The screenshots from left to right each show an individual which is the fittest of its generation (left being the earliest generation and right the youngest generation).

CHAPTER | 10

APPLICATION AND OUTLOOK

10

CHAPTER | 10

APPLICATION AND OUTLOOK

This chapter presents two applications of the developed geometry system. The first application was a Buro Happold internal ideas competition to develop the next generation of modular train stations for Network Rail. This competition took place during the phase in which the implementation of the geometry system into the GA was in progress. The growth and optimisation was working, but the attractor point fitness was not yet installed and consequently there was no spatial control for the surface growth. Example configurations for platform canopies and station buildings were manually assembled, or predefined surfaces were approximated with the approximation tool. The exercise allowed the geometry system to be tested on a target project type, as well as an investigation into which control parameters the prospective user interactive tool had to be fitted with in order to be applicable to this kind of project.

The second application was a design workshop which took place when the geometry system was fully implemented into the GA with a spatial fitness control - the attractor points. Both the surface approximation and the complex surface growth algorithm were beta tested by the participants. The workshop served as a proof of concept, it gave valuable insights into the user friendliness of the tools' interfaces and the usability of the two different design approaches. It also provided an idea of the scope of the geometry principle in terms of diversity in the design outcomes.

10.1. Modular station for Network Rail – case study using the 'Unit Cell'

Network Rail approached Buro Happold to consult the company on ideas for a new generation of modular train stations to be installed all over the United Kingdom, replacing old stations and creating new ones. An internal ideas competition was arranged. Multi-disciplinary groups were formed, each consisting of at least one person from the services, façades and structures groups. The proposal for the modular train station had to include the platforms, roofs and necessary enclosed spaces. One design group was shaped around the idea of employing the Unit Cell principle of this study for the roofing of the platform and service areas.

10.1.1. Constraints

The basic constraints for the station design were that there should be one module for the platforms and the base of the station buildings, and one module for the canopies and the station roofs. It was also desired that the modules could be adapted to individual site conditions, and should be designed to suit single-track platforms, double-track platforms and station buildings that have different requirements. However, the stations should be recognisable as part of a station family.

The key considerations were the speed of construction and low costs through the repeatability of building elements. The challenge was to achieve modularity, while using prefabricated elements to create an individual station design.

10.1.2. Proposal

A family of roof modules was proposed which could adapt to specific station requirements in both plan and elevation. These modules were frames which could receive different cladding materials. The platform and station base consisted of the same box-truss module, and were planned on the same grid as the roof (Figure 165). The enclosed spaces (ticket booths, washrooms, etc.) used custom steel frame modules (CIMC, 2013), which would be fully serviced and furnished off site, and placed under the modular roof. The use of modular elements for most building components created economies of scale in the shop production. The amount of site preparation was reduced by using large structural components in the erection, and eliminating most on-site fabrication and assembly of components, therefore minimising station possession time.

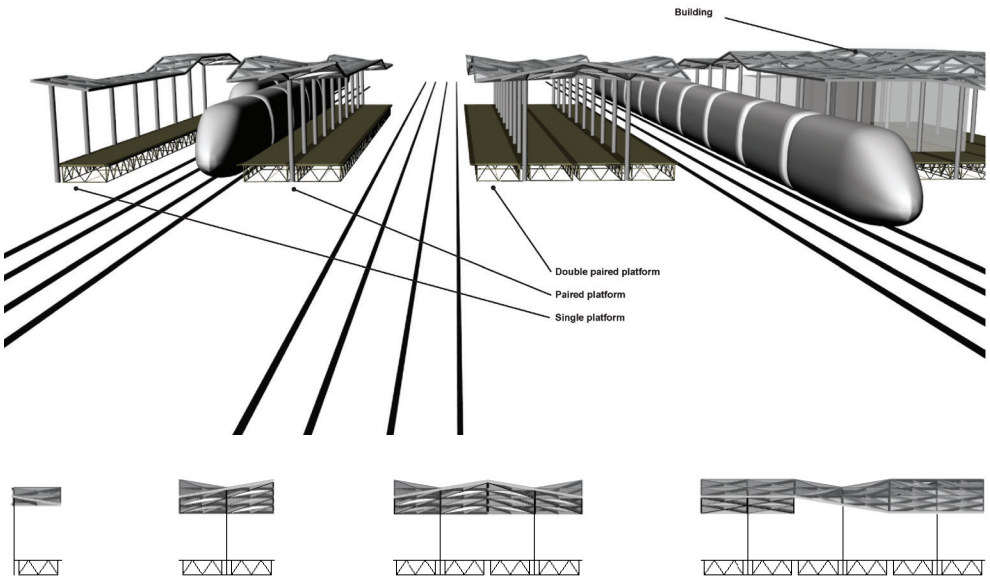


Figure 165
Modular platforms. Isometric view (top) and front view (bottom), showing an example of single and double platforms.

10.1.3. Chassis module

The chassis for the platforms was designed to be made of lightweight pre-welded steel, with the dimensions of 6m × 3m × 0.9m, which would be craned into position. The dimensions meant the chassis could be stackable, and easily transported by lorry or train. The foundation was built of screw piles to minimise the necessary groundwork. Trussing the chassis allowed them to span between screw piles and corners.

To adapt to different on-site levels, a vertical tolerance was achieved with capping plates and screw heads at the top of the piles. The columns were bolted at a distance of 6m to the chassis on site, using a base plate. The necessary tolerance to accommodate the rotation of the column was created using slotted holes. The chassis was designed to receive prefabricated finished cladding modules made from a range of materials (see infill pallet for the roof modules). The size of the platform chassis module also lent itself to being used as a module when designing the bridges which connected the platforms.

10.1.4. Canopy modules

For the canopies, five module types were defined. They were built from triangulated interchangeable panels, which allowed for flexibility in the choice of material. The arrangement of the panels could vary across every station, adapting to changes in site conditions and orientation. Through an analysis of each site, an individual architectural identity could be generated for each station whilst retaining the link to a modular family. The individual roof module was a 6m × 6m timber frame with triangulated cladding elements (Figure 166). Four glulam beams were cantilevered from the central column to pick up the frame corners. Tertiary, thinner, pinned edge beams spanned from the column to the edge beam. The connections were formed with flitch plates, and bolts or pins. The 1.5m × 1.5m triangulated cladding panels were supported on the timber members (Figure 167).

The canopy would have to conform to the following site conditions: sun path, wind direction/strength, views, glare and shelter. The infill panels could be selected from a variety of materials, chosen to suit a particular functional requirement, for cost reasons or to maintain an architectural aesthetic, e.g. by utilising local materials to create a vernacular identity relevant to each station. The material palette for the infill could include glass (single or double glazed), composite—glassfibre reinforced concrete (GRC), glass reinforced plastic (GRP), fibre reinforced plastic (FRP), plastics, stone (natural or reconstituted), concrete, timber or metals.

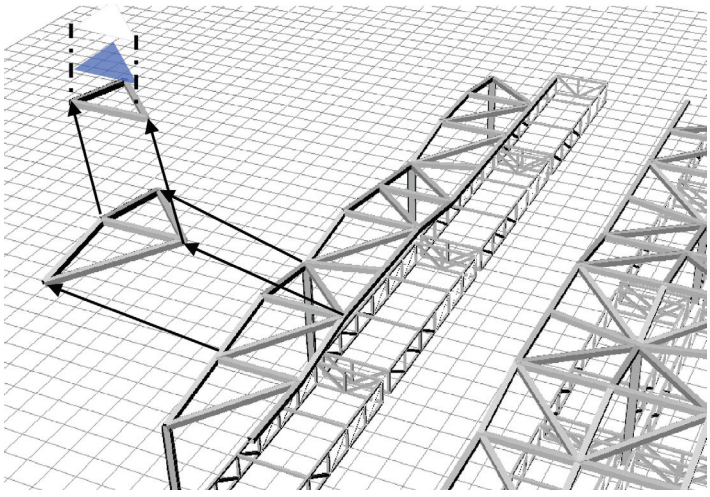


Figure 166
Inter-changeability of various triangulated cladding elements.

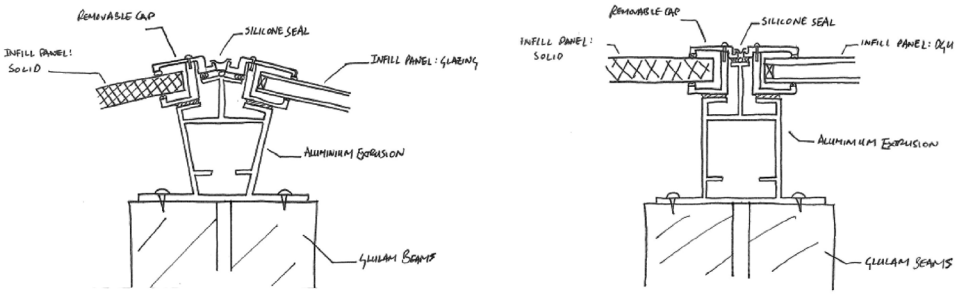


Figure 167
Detailed sketch of inclined and flat panel interface.

10.1.5. Implementation of the 'Unit Cell'

The engineers implemented the Unit Cell concept using manual functions in Rhinoceros and Robot by copying, rotating and assembling modules. The final presented sample configurations for the roof were derived. Only two out of the six group members, one being the author, used the automated random growth of the plug-in which was fully functioning; inputting the parameters for the local element shape, size and global canopy dimension. As a test, the author also modelled a couple of arbitrary canopies and approximated them by applying the approximation plug-in that is described in Chapter 6.

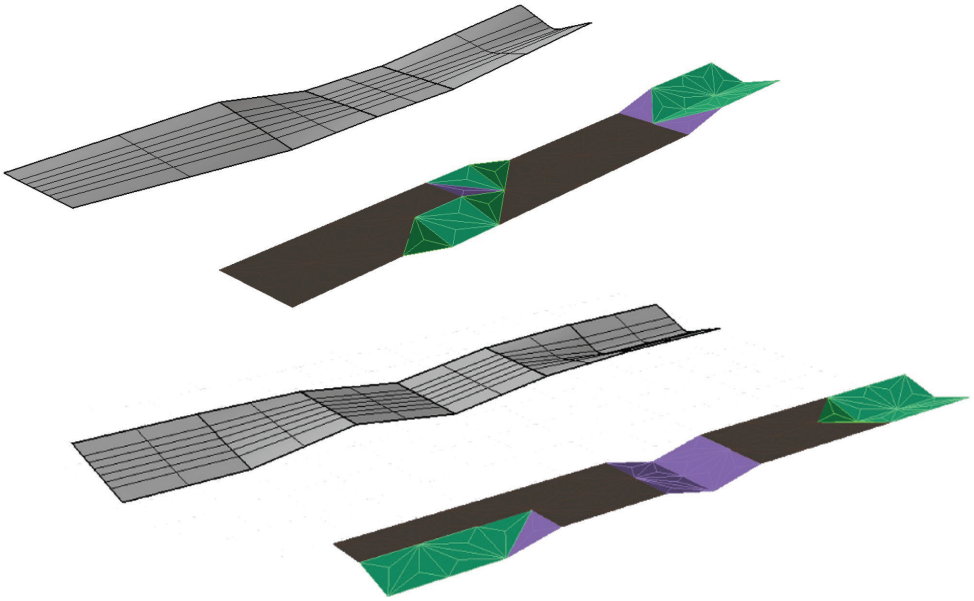


Figure 168
Applying the surface approximation. Two sample canopies and the approximation of those, using three roof module types.

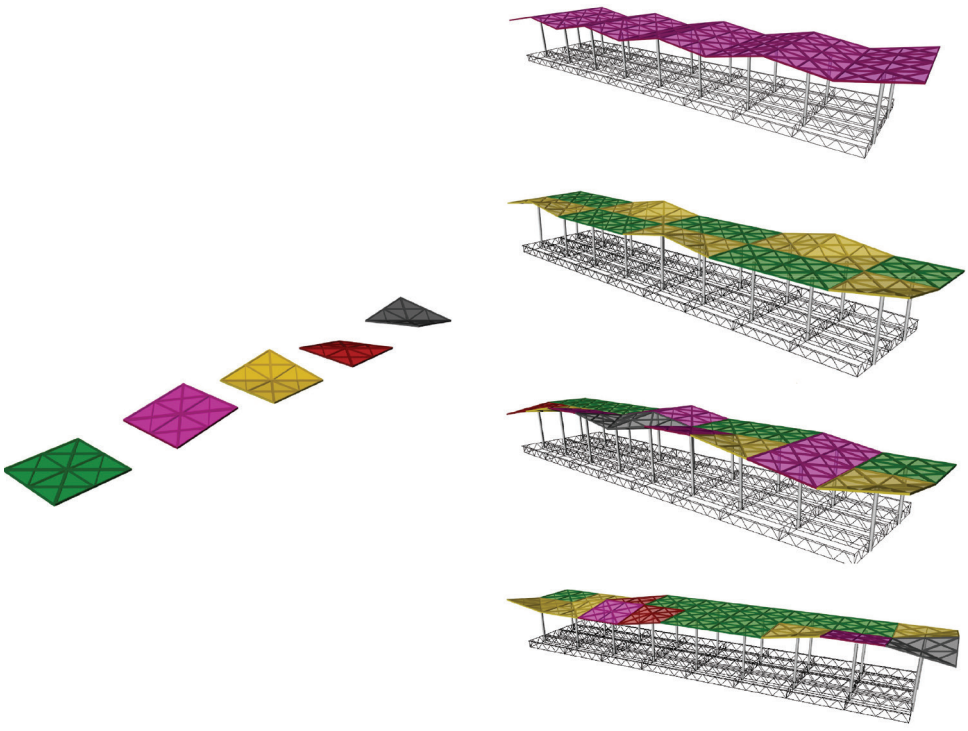


Figure 169
Example configurations of the canopy. Sample formations for a paired platform, using five basic modules or only a subset of these.

Fabrication and erection

The roof module would be shipped to site in two triangular halves. This would be necessary due to the limits of the transportable size. One half would be rigid and fully clad, while the other half would come temporarily braced. The two halves were connected on site to form the full 6m × 6m module. Each module formed a stand-alone element, cantilevering from the ground. As each of the modules was designed to be structurally independent, site tolerance would be considered between the modules, with an engineering gap bridged by an adaptive cladding detail.

10.1.6. Conclusions

The use of a geometrically constrained system such as the Unit Cell lends itself well to serial developments, such as supermarkets, bus stations, train stations and airports. The Network Rail competition offered an opportunity to test the principle of the Unit Cell. The competition took place before the fitness criteria were defined and implemented in the design tool. The application in the competition made it possible to observe what fitness objectives would be sensible user controls. This was discussed in the group; it seemed possible that a simple control through spatial indications could stand in for considerations such as drainage, wind and sun exposure.

It was surprising that the basic square kit of parts was difficult to explain to those not familiar with the principle beforehand. The confusion came from the fact that it was not one module, but potentially five different modules. The engineers played with a single doubly curved roof and column module, and rearranged it so that it resulted in a somewhat interesting articulation. Only in the hours before the day of submission did the senior structural engineer of the group say that he had understood the idea and played with more than one module type. He also then realised that the subdivision of the roof module into planar triangles was neither sensible nor necessary from a structural or manufacturing point of view.

The submitted scheme for the idea competition to design a modular station system for Network Rail employed the potential of repetition and variation of the Unit Cell concept, and showed a variation of roof articulations using parts of or all of the basic modules. They were unnecessarily subdivided into planar triangular fields. The argument for this was that, oddly enough, the system looked more like something that Buro Happold had engineered. The work in the team showed that in-depth engagement with the basic concept was required to fully understand it. This may be because the system steps outside of common modularity, in which only a single module is used, by providing a family of elements instead.

Achievements

- Complete modular scheme provided for station buildings, platforms and canopies.
- Scheme was adaptable to specific site conditions.
- Easily extendable.
- System that allowed stations to be individualised, using a relatively small set of modular units.

Relevance for the research project

- First application of the Unit Cell principle and the program in its state of development at the time.
- Early user feedback for the principle in general and the plug-in in particular.
- Learning about interfacing the roof with, for example, supporting columns, walls and floors.
- Thinking about transportation, erection and maintenance.
- Encouraging the development of the attractor fitness for the plug-in by discussing the functional aspects of the canopy articulation.

10.2. 'Similar and Same' Design workshop

The author was invited to hold a workshop at the University Iberoamericana (IBERO) in Mexico City. The workshop served as proof of concept for both the geometry principle of the Unit Cell and its implementation into a digital user interactive tool. Participants of the workshop, titled 'Similar and Same', beta tested the digital design methods developed during this study. The two methods which were tested were the surface approximation (page 171) and the bottom-up growth of surfaces (page 189). The workshop was sponsored by the Department of Continuous Education at IBERO and two industry partners: Fapresa, a concrete manufacturing firm, and Arqme, an architectural division of Fapresa. It was organised and co-tutored by Pablo Kobayashi, who teaches the 'Digital Logics and Material Expressions' Diploma.

The workshop offered participants an introduction to design tools, both conceptually and practically, by using the implemented methods to produce a detailed design system. The author, in turn, gained first hand user feedback and an opportunity to test the usability of the technology in design practice.

The aspiration for the author was to find answers to two critical questions. The first was whether a variety of designs could be created using this method or whether the geometry system, which forms the basis of both tools, was too constrained and the outcome thereby locked into a particular 'look'. The second question concerned the two clearly distinct approaches to using this geometrical principle; the surface growth and the approximation, and asked whether one approach was more suitable to design than the other.

10.2.1. Workshop participants and structure

The workshop was open to students of architecture in IBERO and external applicants from both education and practice. There was interest from practitioners, but the time, duration and funding for the workshop were agreed only shortly beforehand, which meant interested candidates from practice were not able to attend. As a consequence, all the participants were students of Architecture at IBERO at the time of the workshop, spanning from second year undergraduates to final year of the diploma, except one researcher from the Artificial Intelligence Laboratory at Stanford University and one

candidate who had recently completed her diploma in Architecture.

The workshop was organised into two modules, each lasting two weeks. The first module (08-20/10/2012) covered the introduction to the design processes and the development of a design using the presented methods. The second module (23/10-10/11/2012) concentrated on the extrapolation of the design into a material system. It is not included in this description, because projects are further developed at the time of writing this documentation.

The first module was divided into two phases. In the first phase (first week), both tools and the underlying concepts were introduced and the participants had a couple of days, in turn, to experiment with each tool. In the second phase (second week) participants chose one of the design tools and developed a detailed design system that considered the idea of modularity.

During the second module participants were asked to advance their design by planning and prototyping a real scale system. Ideas suitable to be made from concrete were developed to be prototyped under the supervision of Fapresa. Visits to the Fapresa factory and the Arqme studio gave insights into the manufacturing process for prefabricated concrete elements.

10.2.2. Technical preparation of the tools

In preparation for the workshop both methods, the approximation and the growth of surfaces, were further developed. This development entailed the surface approximation being translated from a C# Rhino plug-in to a Grasshopper component (Figure 170). Grasshopper is a parametric plug-in to Rhino and enables users to construct associative models (Grasshopper). This way the approximation could adapt to a changing input surface, for instance, or the parameters of the approximation itself, such as shape of the elements or the stepping value, could be subject to instant alteration. The Grasshopper Component has four inputs and three output parameters. The main input parameters necessary to execute the functionality are a surface or polysurface (S^1) and a curve grid (C). S^1 is the reference surface to be approximated. C supplies the pattern for the approximation. Additional inputs are a plane that allows the orientation of the approximation to be controlled, the step-size (S^2), which is the same parameter as in the plugin version, and the baking switch (B). When the baking switch is on, the Grasshopper geometry is 'baked' into actual Rhinoceros geometrical objects. The output parameters P, T and M harvest the produced data. The P output lists where a new polyline has been created and mapps it to the corresponding grid cell. T indicates how many different element types are created and how many of each type are used. M lists which type of element is placed at which location in reference to the input grid.

The growth of surfaces method remained a plug-in to Rhino with its own GUI. The GUI received the additional feature that users could monitor and reinstate all individuals of every generation through a tree diagram (Figure 171).

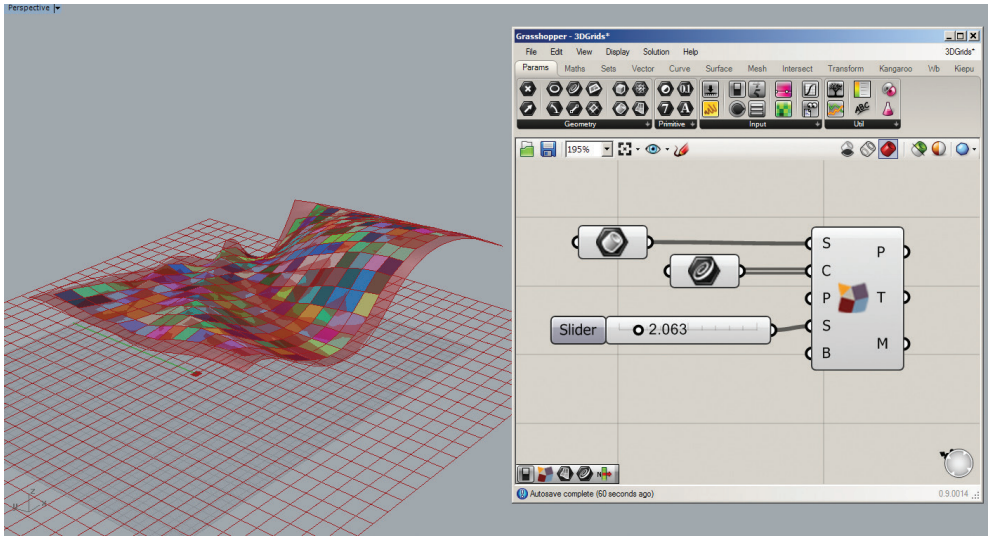


Figure 170
The Grasshopper Component (Kiepu) (right) with the reference surface in translucent red (left).

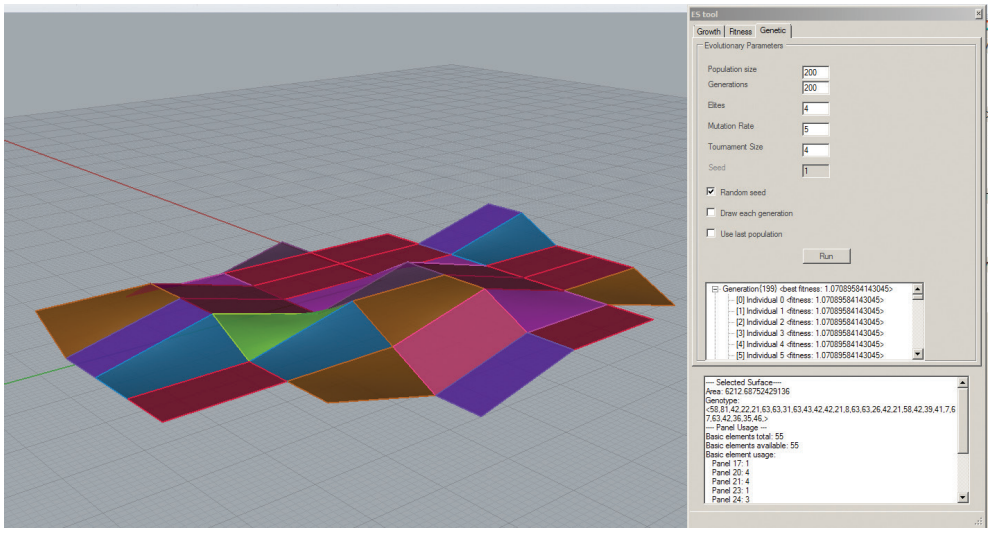


Figure 171
Display of the fittest surface after the last generation run.

10.2.3. Experimentation

Whilst experimenting with the surface approximation, participants were invited to work in an intuitive way and even to try to 'break the system' in order to experience the extent and the limitation of the implementation of the tool, and the method in general.

Participants were given an example file with a simple parameter setting and a freeform surface. During the experimentation phase users focused on operating the grid, plane and stepping parameters. Only a few people changed the input surface (Figure 172).

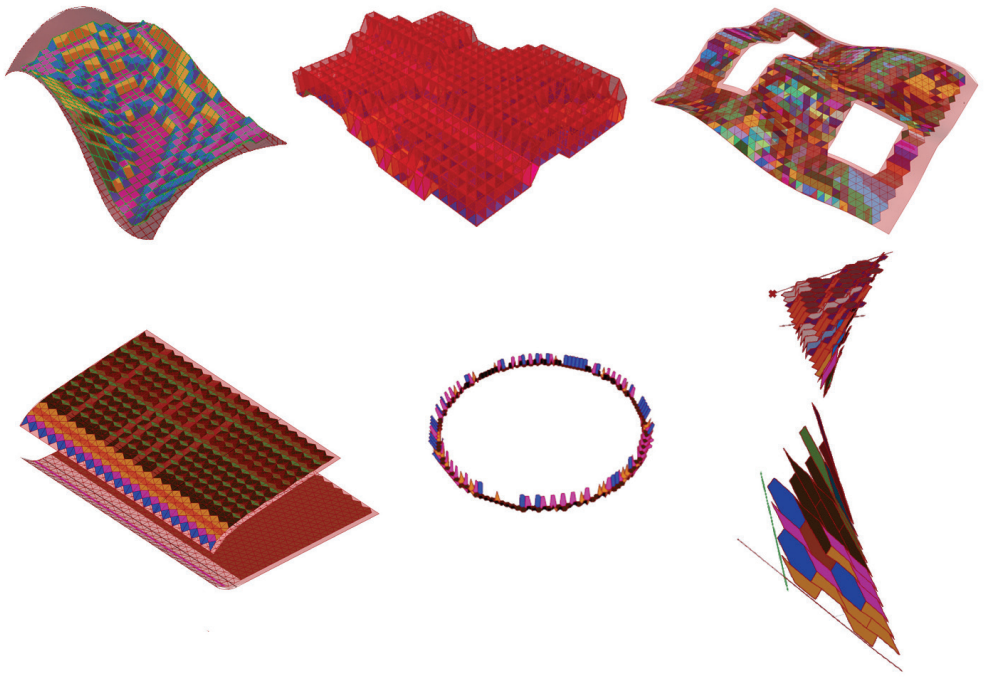


Figure 172
Experiments using the approximation tool with various example surfaces (top) and different input geometries (bottom).

For the second experimentation phase a strategic approach was recommended, as the tool is far more complex. In order to understand both concepts, of bottom-up growth and top-down optimisation, that are complimented in this single plug-in, participants were advised to only change one parameter at a time. The rigorous setup of experimentation allowed users to comprehend the two levels of control: firstly the growth parameters which determine the shape, size and number of the local geometries and the size of the resulting surfaces, and secondly the genetic parameters which control the operators of the evolutionary mechanisms.

Participants quickly found that the growth always started at the coordinates x, y, z = 0. They needed to make this constraint comply with what they wanted to do. It is an obvious shortcoming of the user control, yet once participants had understood this restriction they worked their way around it. By the end of the workshop most participants had tried out and understood all the parameters, including the evolutionary operators, except the tournament selection. This parameter was in effect not understood by anyone. As it is a crucial operator it could be automated so that the default setting adjusts itself in a particular ratio to the population size. Participants learnt that they needed to match the geometry parameters with the attractor fitness so that it could actually influence the result. Some unexpected results occurred when these two factors mismatched, see Figure 173. The surface on the left is built from a set of three triangle types. The second surface from left illustrates what can happen when attractor points are located away from the starting of the growth, which at the time always started at the centre of the surface at the coordinates x, y, z = 0. The two images on the right show a mismatch of the attractor location and the setting of the geometry parameters.

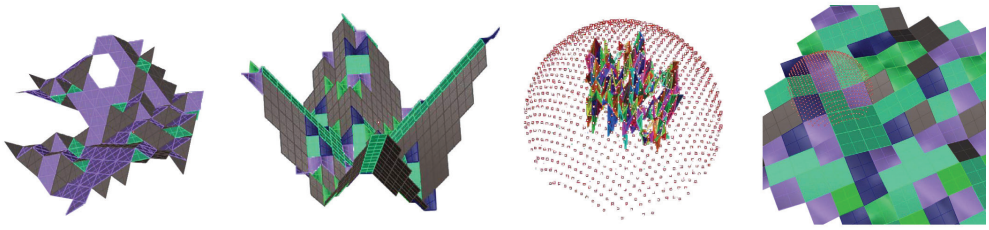


Figure 173
Examples from the experimentation with the surface growth plug-in.

10.2.4. Design development

Whereas, in the experimentation phase, mixed and more elaborate tessellation patterns were defined, in the design development the tendency moved towards the use of simple shapes such as squares and triangles with the aim of reducing the number of different types of components. The two representations, surface or frame, were used in equal measures among the participants to create their designs. Seven of the resulting projects are described below, each is an independent design system; the models can be reconfigured to create different configurations using the same parts.

Model A consists of six surface element types. Their base shape is the square. The digital elements served as placeholders for the more elaborate stepped description of some of the component types (Figure 174). A stepped profile continues across the edges of neighbouring elements. Thinking of acoustic panels, the author envisaged the control of sound wave reflection by locating peaks, valleys and flat areas in reference to acoustic needs, while the overall articulation remains in one plane. Model B consists of a family of four different triangular shapes which were literally translated from the digital into the physical model (Figure 175) using a specific design brief for a glass-steel canopy. Model C combines frame and cladding elements using three generic frame module types, which can receive different surface definitions with diverse functionalities (Figure 176). Model D exhibits a single generic node, connecting the two different types of rod elements (Figure 177) and Model E is made using a hexagonal pattern and a high distortion, through a large stepping value. Four types of frame elements were produced and assembled into the presented configuration (Figure 178). Model F is focused on the construction of the nodes. The number of element types was reduced by subdividing each node into two rib sections. These ribs slide together at mid-length, forming a node with four rods, and each rib section repeats over the other node types. Work also focussed on the design of a connection between node elements along their rods (Figure 179). In Model G the approximation tool was applied on top of a surface previously generated with the growth process. By doing so this system is repetitive on two scales: the three larger module types and the cells which create the local articulation of each module (Figure 180).

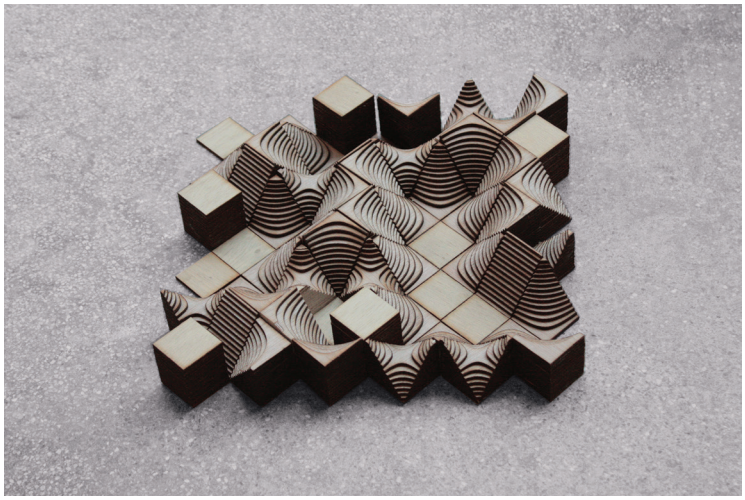


Figure 174
Model A is made up of six repeating element types. The author of this model is Teddy Nanes.

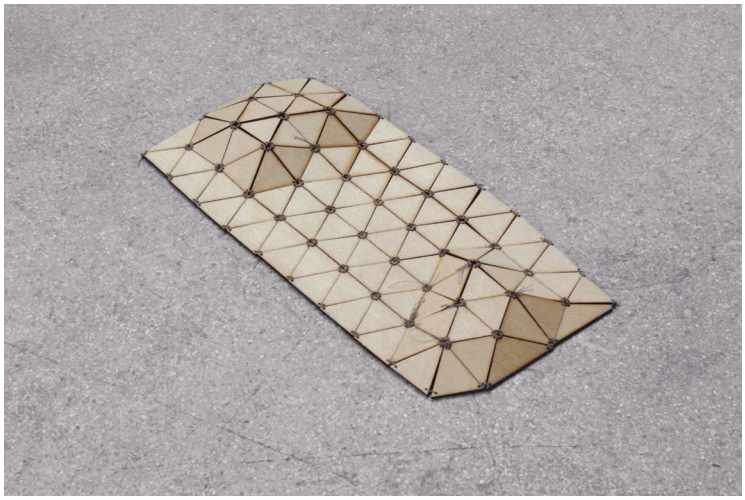


Figure 175
Model B by Francisco Javier Regalado Abascal consists of four different triangular shapes which are directly translated from the digital model.

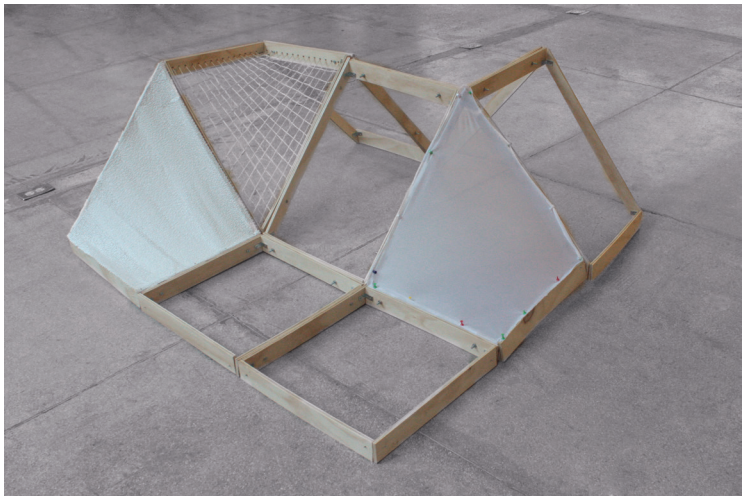


Figure 176
Model C by Jose Cherem comprises a combination of frame and cladding elements. There are three generic frame module types that can receive different surface definitions for various functionalities.

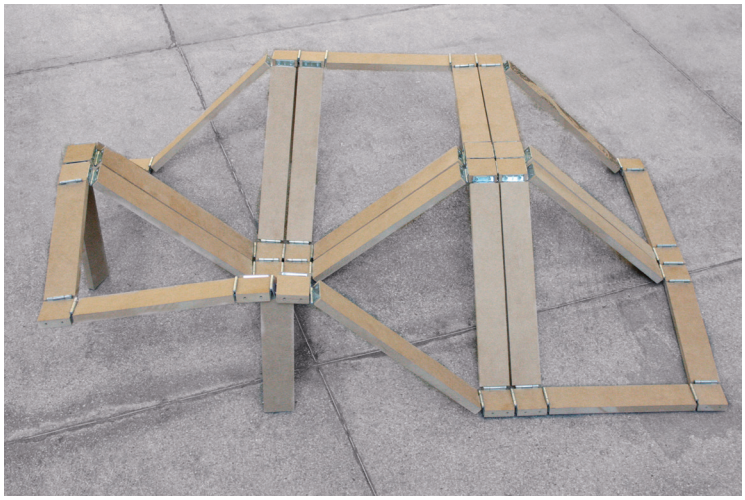


Figure 177
Model D is made up of a single generic node that connects two different lengths of rod elements. The author of this model is Rosa Pintos Hanhausen.

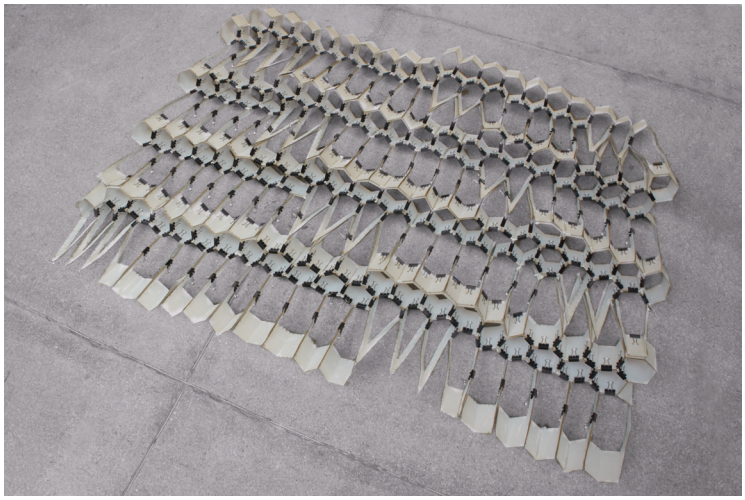


Figure 178
Model E consists of four repeating hexagonal frame types. The project was developed by Elias Kalach Hanano.

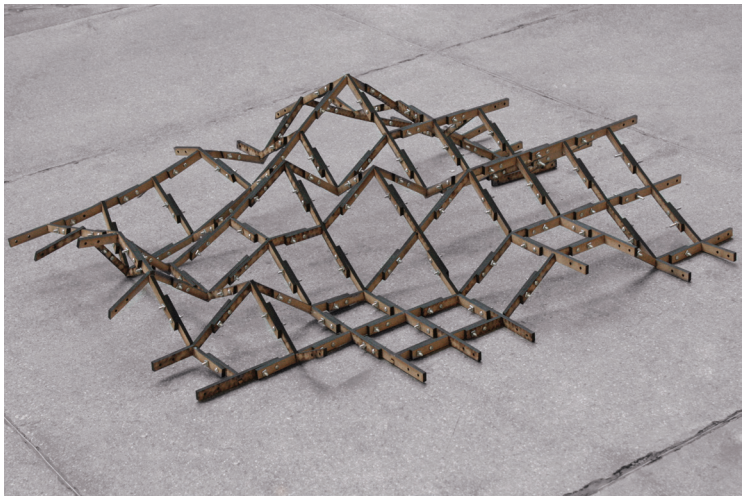


Figure 179
The work on Model F was focused on the construct of the nodes and the connection between them. The author is José Paulo Péres Lemus.

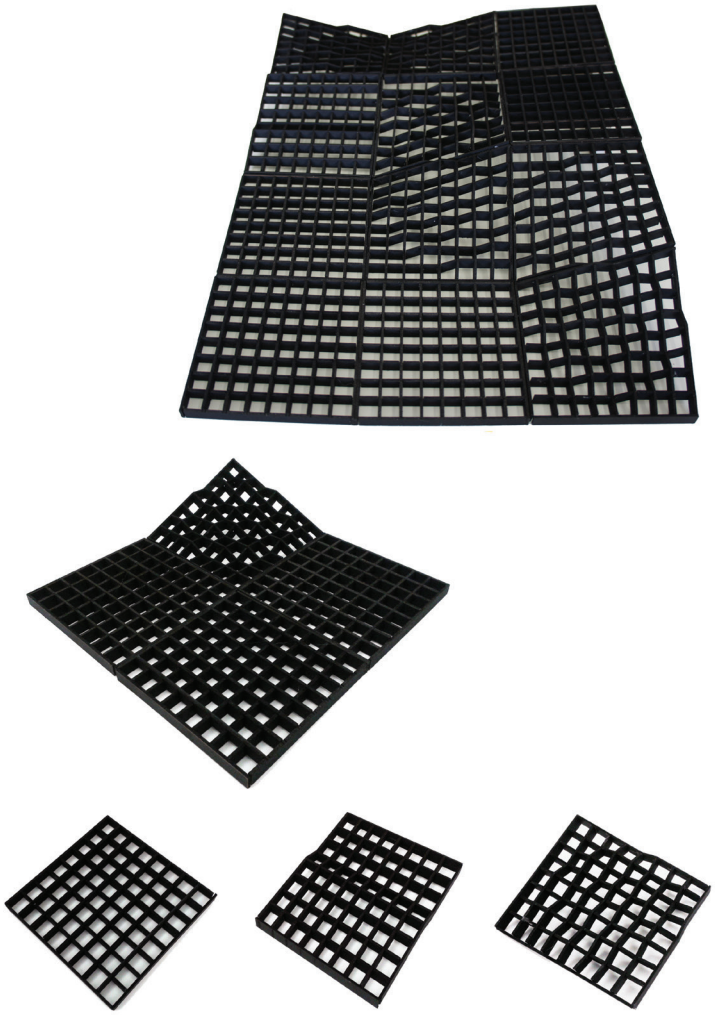


Figure 180
Model G resulted from applying one tool on top of the other one. This project was developed by Francisco Villalon.

10.2.5. Conclusions

The assumption prior to the workshop was that the fast, responsive top-down approach to approximating an input geometry would gain greater popularity than the conceptually far more complex bottom-up process of growth. In the latter the geometry is truly 'generated', which means that the user has to set up the whole framework to create a model, defining parameters for the local geometries and generating the kit of parts and spatial control points to guide the articulation of the surfaces to be grown, as well as driving the evolutionary operators. The number of interrelated parameters is large and therefore the feedback and control slower, not so much in computational time, but in the sense of design output.

It therefore came as a surprise that both approaches were used equally and that in the final feedback from participants the majority enjoyed generating a kit of parts and initiating their design exploration with this constrained set of building elements. They also liked the ability to view the development of surfaces over the course of generations and receiving alternative surface solutions to the same framework setup; both are features of the evolutionary technique. Participants said that at first they preferred the easy use of the approximation and the flexibility of the Grasshopper environment,

but once they understood the different levels of control in the growth method they appreciated that it initiated a different, decentralised and strategic way of thinking.

One observation was that the approximation technique, which was conceived as an optimization, since it is the reverse-engineering of an input geometry, turned out to be suitable for intuitive experimentation. This is due to the fact that the geometry logic is plugged onto any given input geometry. Hybrids are created, as the logic of the geometry system is forced onto the independent geometry logic of the input surface.

On the other hand the less deterministic growth process, which was conceived as a conceptually driven design approach, turned out to provide greater control and therefore a more strategic design development.

The results of the workshop in the form of design models confirmed that there is extensive design scope, despite, or possibly because of, the constraints the method dictates. Each result is an independent design system, in that every project consists of a number of modules which can be reconfigured to form different configurations. Some projects are literal translations of the digital representation; the focus here was on the design of a connection mechanism. Other projects extended the appearance of the local geometry of the component.

There is scope for investigating what the implementations are and which parameter bears greater scope for varying the resulting design, the pattern of the tessellation of the componential surface or the complexity of the individual local geometry. The fewer the number of parts involved in the generation process the more recognisable the system-nature of the method becomes.

10.3. Outlook

There are two areas of the building industry which come to mind, where the Unit Cell geometry concept and the tool to design surfaces can be employed. The first is in highly modular projects, such as the Network Rail example. This field comprises all kit designs, such as for supermarkets, bus stations, train stations and airports. Modules for these schemes are employed over a number of structures, rather than a single one. Because of the scale of such schemes, components can be standardised. They can be prefabricated, which can reduce their cost and improve their performance over time. It can be presumed that the designs in such applications would be relatively simple, assuming that the number of module types would be small and the base shape of these components and their articulation within the component boundary would be relatively plain.

The second area of application is that of single bespoke structures. Here, the repetition of building components within the same structure means the individual module type can have a more complex design. That is, the shape outline and the articulation within the element frame can be elaborate. Through the reuse of formwork and repeating manufacturing paths, the project can still be feasible. Both scenarios are examples of what is known as ‘mass customisation’, where cost-efficient yet individualised designs can be created.

The presented projects from the workshop at IBERO show in the conceptual state the design potential entailed in repeating sets, which can be envisaged for both large modular building schemes or repetition in a one-off structure.

CHAPTER | 11

DISCUSSION AND CONCLUSION

11

CHAPTER | 11

DISCUSSION AND CONCLUSION

11.1. Discussion

"Should a tool make design easier? – Perhaps it should make it more thoughtful." (Aish 2011)

Aish suggests that digital tools can help users acquire knowledge that otherwise would not be accessible. His point of view is supported by Coates' statement, cited at the beginning of this thesis, that the computer can be used beyond drafting, as a means 'to think with' (Coates 2010, p.26).

In this respect Steadman argues that "The way in which such studies are applicable and useful to the design of new buildings is in providing increased knowledge and greater understanding of the particular aspects of building geometry or behaviour in question" (Steadman 2008, p3). The author agrees with this proposition; the task of design is not taken away from the designer through digital design processes, rather they aid the acquisition of knowledge necessary to respond to the task at hand.

Steadman believes that such understanding does not necessarily evoke new theories or discoveries, but expands the practical experience of the designer.

Here, Steadman makes an important comment on the subject of completeness, and subsequently the application of design methods, by saying that "The knowledge could be built up as in a piecemeal and gradual fashion, as in present-day building science, without any necessity of being immediately all-inclusive or complete" (Steadman 2008, p3). The author interprets this to mean that there is a new independent field of architectural investigation which continuously develops. However, architectural research should not be misunderstood as another science (Steadman 2008, p. 2).

Supporting this opinion, the author believes that however small or large the scope of the design method is, or what aspects of the build design it covers, it should provide a well understood framework within which solutions can be explored. This framework can be expanded and improved, or eventually replaced. The creator and the user of the system are equally involved in the progress of the field of architectural investigation into design methods.

As referred to in the Introduction to this thesis, Steadman sees the potential in design methodology for architecture in the integration of not only functional objectives (those that are performance-based), but also those which are regarded as 'architectural'. One he describes is the geometrical organisation of parts and structures (Steadman 2008, p2).

This thesis recognises the necessity of studying the relationship between the global and local geometries of existing organisational structures, which creates the opportunity to propose a design tool informed by real-world parameters through the employment of discrete parts. The tool prototype of this thesis enables the controlled exploration of surface solutions embedded in a clearly defined framework.

11.1.1. The making of generative design processes using discrete geometry

Once the potential of generative design processes that use discrete growth models has been recognised, the question remains of how to actually create them. In particular the definition of a novel growth model calls for some kind of innovation, which in turn would explain why there are few existing examples.

One way to originate a geometry system is through the study of existing geometric organisations, and the use and adoption of models which mimic these organisations. Such systems are found in the sciences, for example at the intersection of biology and medicine where models are developed to simulate the growth of viruses (which are crystal-like structures) or materials science, where the behaviour of specific aggregates is mimicked.

Interesting early work has been conducted taking advantage of the latter example. Existing models that imitate aggregate behaviour, such as 'Discrete Element Modelling Simulation' (DEM), are borrowed from materials science and aimed at being adopted for architectural purposes (Dierichs et al. 2011). Aggregates are large numbers of elements in loose contact. The formation of aggregates has aesthetic and functional features, relevant to the design of structures, which are yet to be explored. The interesting aspect is that such structures are uncoupled from a design surface (Figure 181).

Figure 181
Rendering of an aggregate formation resulting from a DEM-simulation (ICD-ITM, University of Stuttgart, 2011).

However, this is different to the investigation in this study, which looked at geometric formations with explicit connectors and joints. Various geometric experiments were conducted, including trials which (in the context of this thesis) could be referred to as 'naïve'. They were complimented by the use of an established model which is the static configuration of closely packed spheres. The model was extended so that continuous networks of adjacent triangles could be defined. Variations of the network led to the concept of the Unit Cell growth model.

The study and adaptation of established models which mimic how existing geometric organisations form, seems to be a suitable approach to create novel growth models for generative systems. It may be the only strategic way to do so.

It is then that the making of design processes is likely to occur in research, rather than in industry or, as in this case, at the intersection of the two.

The challenge here, as in every instance of architectural design computing where a model or a simulation is borrowed from an entirely technical and scientific context, is the adaptation to architecture itself. The danger lies in justifying the results of such an

approach by the fact that an established model was used, without ensuring that the measures of verification are adapted, together with the transfer of the model into a different application context.

11.1.2. Adaptation of generative processes in design practice

Another interesting question is how inductive design processes with discrete geometry will be adopted in practice. There has been a revival of modular design, and projects such as the Queen Alia airport in Amman by Foster + Partners have been positively received. Popular existing analogue design systems, such as Eladio Dieste's reinforced brickwork system (constrained product) or Antonio Gaudi's constrained geometry (constrained modelling), form a good basis for the study and development of computational processes with discrete representation of geometry. What the examples have in common is a wide catalogue of possible solutions. However, an underlying fear might still be that designs could be too easily recognised as outcomes from a particular tool. It appears to require courage to recognise that the desired visual complexity does not need, and possibly does not derive from, complicated models. It is also evident that present modular designs are often reconfigurations of the same parts and do not make use of the potential that computational geometry systems could offer.

11.2. Contributions

The contributions to knowledge this thesis makes are listed below. They are divided into two groups. The first group describes those concerned with the identification of the current state of practice. The second group comprises the design implementation based on the findings in practice.

Contributions to understanding the state of the art

- The clarification of the distinction between different types of intricate form and what they mean in relation to their global and local geometry. These types are freeforms, complex forms and hybrids.
- The recognition that the definition of the local parts which constitute a global geometry, in retrospect, is a multidimensional problem. These aspects are the two-dimensional outline, the shape of the local element, its articulation in three-dimensional space, its physical offset (which represents its own material thickness or supporting structure) and the interface to neighbouring parts. The interface comprises the description of the edges, and necessary connectors along the edges or nodes.
- The identification of recurring problems in the design and engineering of elaborate architectural surfaces. These are first, the readability of the model (the clarity of geometry description), that is necessary for the transfer between applications, analysis, post-processing and detailing, and second, the definition of local building parts. This includes the problems described in the multidimensional tessellation task.

Contributions of the design implementation

- The conversion of the identified problem pattern in the design and engineering of elaborate architectural surfaces, into constraints suitable for an evolutionary design system.
- The development of a novel growth model, the Unit Cell, which overcomes the hurdles associated with creating continuous surfaces from discrete elements through a bottom-up approach. It provides a discrete representation of buildable parts combined with parametric control of repetition. The representation of the parts as frames or surfaces can be placeholders, i.e. only the boundary must be maintained. Their physical offset is known and the connections to neighbouring elements are resolved. Through their discrete description they are easily edited and transferred between different applications.
- The implementation of the Unit Cell growth model into the program structure as an 'element factory', which allows the generation of kits of parts under user specifications, such as the type of shape, local element size, global surface size and level-stepping.
- The development of a new representation: Best Fit mapping. The mechanism maps the genotype into the phenotype, ensuring that the crossover operator of the Genetic Algorithm can function properly.
- The definition and implementation of fitness objectives, i.e. the size of the created surface area, number of created gaps, closeness to attractor points and undulation of the surface.

11.3. Contrast to the reference projects

The design tool prototype, developed as part of this thesis (Figure 171), overcomes the limitation of the referenced ECSS system, where the growth model is a hard-coded set of five surface elements. The tool provides a concept which allows the evaluation of different polygonal shape families according to the user settings. The model is implemented for three- and four-sided polygonal shapes as a proof-of-concept. The user specifies the features of the local geometries (type of shape, the representation, stepping and the relative size) and the 'element factory' generates the corresponding family of parts. The catalogue of possible surfaces which can be built from these parts is thereby much larger than that of the ECSS tool. However, akin to the ECSS system, every solution can be built. Rather than limiting the number of local parts, the quantity of form types can be controlled. Once the user executes the program, the initiated set of local parts remains the same during the course of generation cycles. Over this course, only the assembly of these components is altered and optimised.

In Genr8, the optimisation cycle takes place on the grammar level, but only if the evolved grammar is chosen. This means that the parameters that the grammar consists of are automatically optimised and are not part of the user control. This results in a change in the surface articulation in reaction to the environment while it grows and also in a change in the local geometries that these larger entities are built up from. In Genr8 the user has no control of this grammar, once selected. Even though

in its current setup, the optimisation of the grammar in Genr8 is hardly discernible, the idea of optimising the local geometries remains vital. When reflecting on the possibility of transferring the idea of grammatical optimisation onto the growth model of the presented Unit Cell project, it seems sensible to use a top-down approach akin to the approximation exercise, where a given geometry has to be described with a controlled number of element types. Major design decisions in a top-down tessellation process are already made in the form of input geometry.

This is a more critical question for the generative process, where a main part of the design authorship lies in the definition of the geometry parameters. There could be the option that the user selects a subset of the growth parameters, rather than all of them, to become subject to the combinatorial optimisation. This means, for instance, that the user could fix the type of shape to preserve a particular pattern, while its size and stepping would be automatically optimised. The aim of an interactive design tool, however, must be to maintain the creative control of the user.

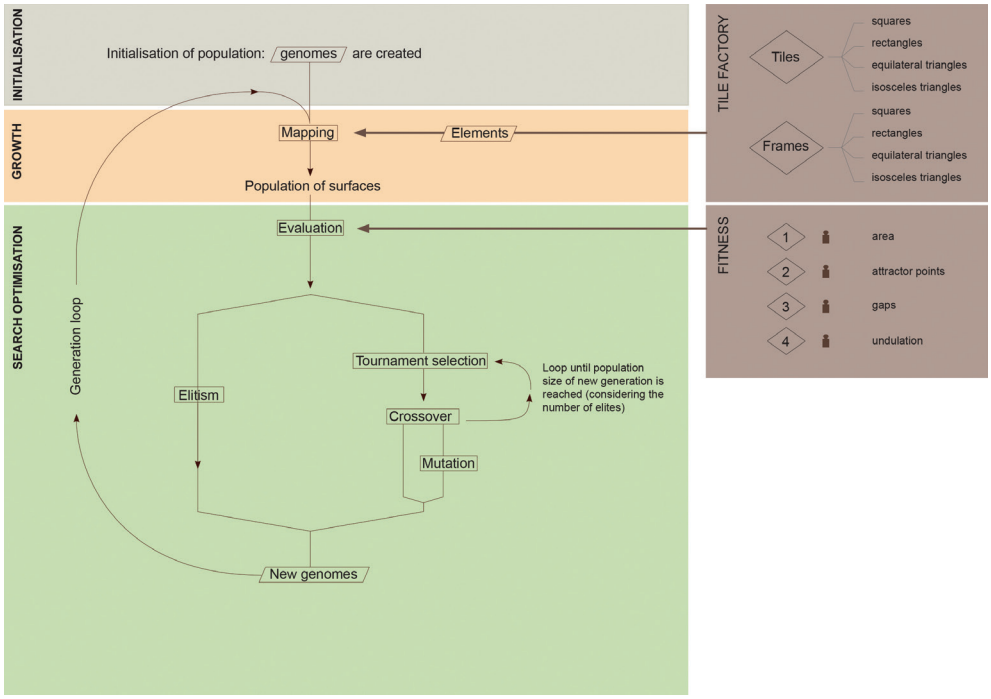


Figure 182
Architecture of prototyped design tool.

11.4. Recommendations for further work

A number of recommendations and ideas are described below as potential developments of the design project.

11.4.1. Recommendations for the 'Unit Cell' growth model

A suggestion for the further development of the Unit Cell growth model is to include the stepping variable in the x- and y-directions, in addition to the stepping already implemented in the z-direction. The surfaces could then exhibit differently sized elements and, in this way, increase the possible adaptation to geometric or performance requests.

A further recommendation to extend the list of growth parameters would be to include the choice of higher dimensional polygonal shapes. A mathematically elegant challenge would be to extend the mathematical formula which calculates the number of elements for every given setting. To be complete, the formula should include the filtering of geometric repetitions. This generalisation, however, is not necessary for the applicability of the tool.

11.4.2. Recommendations for the growth process

The rhomboidal growth order allows the creation of closed surfaces and supported geometry variations. However, an additional parameter could be the choice between different growth orders. More rigorous tests could be conducted to compare the different growth orders and their influence on the generated form. Another test sequence could be carried out to observe the effect on the resulting geometries that are produced by the starting position and element type of the growth, known as the seed.

11.4.3. Recommendations for the user control

All growth parameters could be employed, either as control parameters, or as part of an optimisation routine. The first variation could be that the user explicitly specifies the growth parameter settings, and thus decides which geometry set and order of growth is utilised to create the surface designs. This could be as detailed as required, because the user could decide to employ the whole family of parts or a subset of it.

The second variation could be to incorporate an optimisation cycle in the growth parameter settings. In this model, a search would be conducted of combinations of growth parameters (including the shape, size, stepping and growth order), to best match the user-defined fitness. Both approaches could be part of the same system, so that the user has the option to either control the growth model or hand it over to an automated optimisation course.

11.4.4. Recommendations for the fitness objectives

The discrete description of the parts does support easy post-processing of the solutions and transfer to other application types. It could also be imagined that the discreteness would be utilised to extend the objective measures, by incorporating performance simulation as a fitness objective, either as an integrated analysis or linked to an external analysis package. However, this is a separate extended discussion. The performance simulation for a preliminary comparative analysis can partially be substituted through pure geometric measures, such as the distance between points of support or span over height ratio (Hemberg, O'Reilly 2004).

11.4.5. Recommendations for the evolutionary operators in the genetic algorithm

The extended test of the genetic operators showed that the optimisation worked well. The tests were conducted on a specific problem size and with a single fitness criterion, the area of the surface.

The test also showed that the operators (elites, mutation and tournament size) had a large impact on the performance of the optimisation. It is relatively easy for the user to comprehend the fact that a high number of individuals in a population create a wide and thereby advantageous gene pool. It is also straightforward to understand that a high number of generation runs is beneficial, as the solution pool can be searched exhaustively. At the same time, it is desirable to save computational expense by not running the process beyond the point where the results converge. Nevertheless, it is difficult for the user to estimate which size of tournament, mutation and elitism is the right one for each individual problem. Therefore, it would be beneficial to provide the user with automated suggestions for the setting of the genetic operators, according to the problem size. To include these automated suggestions further tests would need to be conducted. Different problem sizes would need to be observed, the fitness changed and multiple fitnesses potentially employed at the same time. This would then allow a generalisation of the relationship between the problem size and the necessary setting of the evolutionary operators.

A more substantial change in the implementation to decrease the computation time, would be to consider reducing the problem size and, in this way, the number of variables. In the current representation, each element in the surface is represented by one gene in the genome string. The size of the surface therefore corresponds to the size of the problem. One change that could be made is that a single gene would represent a number, not just a single element.

A fundamentally different approach would be that the surface is represented by a function. The indices of this function could be subjected to the optimisation. This would eradicate the employment of a growth model and the literal process of growth.

11.4.6. Recommendations for the surface approximation

The surface approximation was initially set up to only test the scope of the Unit Cell growth model by applying its concept to the approximation of the given surface geometries. However, the application can be used as a tessellation tool when reverse-engineering real project geometries, by controlling the pattern and size of the geometry sets employed.

For this purpose, a number of improvements are suggested here. It would be important to include the control of level jumps, to be able to manage the number of different parts used for approximating the input surface. This is the control of level stepping (currently only in the z-direction), which regulates the z-directional level differences between neighbouring elements and between the points which form one element.

At this stage, each point that is created at the surface and vertical grid line intersection jumps to the nearest point in the three-dimensional matrix, independently of where the neighbouring point is located. This results in many different tile forms being created and there is no control of their number. However, a consequence of being able to constrain the number of level jumps is that, if the allowed stepping is low, it almost certainly causes a large deviation from the input surface. This leads to the criterion of close matching of the input surface. The procedure does not measure the deviation from the input surface yet, which is desirable when the aim is to match the given surface as closely as possible. Therefore, the inclusion of a deviation measure

is suggested. This could be managed by an optimiser that operates the growth model parameters (shape, size and stepping). Different parameters could then be specified by the user or subjected to the optimisation process.

11.5. Experience in practice

There was a temporary aspect to the service that the GG group provided. The creation of the group coincided with a boom in geometrically intricate design proposals and the need in Buro Happold for specialised modelling skills, to both analyse and detail these types of form.

The initial scope of the work embraced the visualisation of suggested structural systems or the illustration of construction sequences. Gradually, the range of work extended into providing design suggestions based on the computational modelling skills and theoretical background of the members in the group. The most significant part of the work can be classified into two types: the pre-rationalisation and the post-rationalisation of an intricate form.

Pre-rationalisation means that the group was approached at the beginning of the design concept. In design workshops, together with the client, it was decided which functional and which aesthetic factors would drive the project. This led to a number of collaborations in which computation played a key role and was employed to initiate the design. Even though ideas of structural form are deep-rooted, it was not common to use automated design strategies, such as inverting structural deflection as a form-giving factor (digital form-finding) or applying thickness according to the structural performance requirements of a design surface. The approach was considered fairly experimental. This is perhaps the reason why some architectural practices, known for their experimental rigour, were interested in pursuing this path.

More recently, integrated design (which informs design by performance measures) has become more popular and present in practice, where parametric models are connected to simulation applications. Geometric features are then linked to their performance measures, which are monitored by analysis tools that simulate structural or environmental behaviour (heat, wind and sound). To connect different applications with each other is an efficient way to speed up feedback loops between factors that influence the design progression. An effort is made to incorporate all of these decisive functional measures into one framework - the notion of Building Information Modelling (BIM).

The group's other role in projects was to help as consultants on geometric issues that occurred after the conceptual phase passed, but had to be solved to allow for the analysis and the detailing of the project. These tasks are identified in this thesis as post-rationalisation, and they formed the majority of projects that the author worked on. The main post-rationalisation task was to find out where a problem originated and to propose alternative routes to tackle it. That meant either eliminating the problem at its source, which often meant that a model had to be remodelled from scratch, or trying to fix the immediate problem. The group always suggested returning to solve the initial problem (as long as it did not interfere with the geometric intent of the design), rather than having to fix accumulating difficulties. This advice was often, but not always, taken on board.

The increasing popularity of parametric software initiated a less arbitrary use of geometry, as these applications demand a certain geometric rigour. However, there is still large scope for the design community to establish better geometric knowledge of the design and handling of intricate form, as is evident through the projects documented in this thesis. This will continue to happen through consultation in the short term and education in the longer term. The GG group's work mainly attempted to increase geometric awareness and influence designs using functional measures, wherever this was possible and where the interest was embraced.

The research in this thesis promotes a pre-rationalised approach. However, the main contribution of the study was to projects where post-rationalisation was demanded. The software tool documented in this thesis only produces pre-rational designs, but the findings from its development largely influenced the work on post-rationalisation.

The in-depth examination and documentation of project work necessary for this research contributed to the expertise that the group developed. It enabled the classification of global geometry proposals, and aided the diagnosis of the origin and nature of problems. The main focus of the research, into the relationship between the global and the local geometry, supported the advice to the client as to which rationalisation measure was appropriate for the given geometry. Instead of zooming in and fixing a problem locally, the study enforced zooming out and putting the problem in a wider context.

For the study, it was necessary to spend more time on some projects that were immediately relevant to the research. That meant that apart from normal involvement in those projects, the author also revisited or shadowed them. Interesting findings and potential contributions were discussed with the group and, if applicable, introduced to the client.

11.6. Conclusion

Reverse-engineering architectural surfaces is one way to make use of computational resources. This is a centralised approach, where the design is understood to be created when the global form is described. Smart processes have been developed, and continue to be, to tackle the challenge of discretising elaborate form in retrospect.

Perhaps it is thanks to the designs which were conceived without considering how they would be realised, that the vast progress in smart technological methods and manufacturing techniques in industry was made.

However, this approach seems to split the process of design into two separate tasks, the conception of form and the engineering of form, which could be brought closer together rather than pushed further apart.

This research recognises an opportunity for an alternative way to make use of computational resources and to thereby overcome recurring problems in reverse engineering. This is by setting up and deploying decentralised design processes with a discrete growth model. Such processes allow for the integration of explicit considerations to do with design realisation during the design conception. A generative system that employs clearly described geometry enables the controlled exploration of the catalogue that is contained in its geometric framework.

The study presents a design tool prototype of such a method, which combines bottom-up growth of architectural surfaces with top-down evolutionary search and optimisation.

The method uses a novel growth model that employs a discrete representation of buildable parts, combined with a parametric control of repetition. The system overcomes the hurdle associated with creating continuous surfaces from discrete elements in a bottom-up approach. It is a response to Ball's suggestion that form 'could not be grown in a naïve way' (Ball 2007).

The representation of these parts as frames or surfaces can be used as placeholders, of which only the boundary must be maintained. Their physical offset is known and the connections to neighbouring elements are resolved. Through their discrete description they are easily transferred between different applications.

Every family of parts generated under user-specified parameters is a buildable set of parts and thereby every grown assembly of these parts is buildable too. Referring to Doscher's distinction (Doscher 2006) between the local and the global geometry of an intricate form, here they are coherent and of the same kind.

The emphasis and motivation for the growth model is that surfaces can be designed using a parametric kit of parts. Relating to the notion of emergence, it is an attempt to create complexity from a simple set of components and simple rules of local interaction.

The growth model is imbedded into a GA to provide a selection mechanism that helps the user to find good surfaces in the solution pool. The operators of the evolutionary engine are tested to ensure the selection and optimisation is working. The growth model principle was applied in a competition project to examine its applicability in real-world employment.

This thesis promotes the use of computational methods to advance the design and buildability of integrated architectural surfaces, and will hopefully contribute to the discussion, development and application of digital design tools in practice.

REFERENCES

REFERENCES

AISH, R. 2011: *DesignScript - a domain specific end-user programming language.* Key Lectures. Design Modelling Symposium Berlin. Academy of Arts Berlin, 12/10/2011. Available online at <https://www.design-modelling-symposium.de/frontend/index.php?page_id=967&ses_id=d72cafc9cfb81185444f998ec89eeee5>, accessed 22/05/2013.

ALEXANDER, C. 1964: *Notes on the Synthesis of Form.* Harvard University Press.

BAILEY, B., RAICH, A. 2006: *Effects of User Preference on Multi-Objective Roof Truss Optimization.* In GECCO '06, Genetic and Evolutionary Computation Conference. User-centric Evolutionary Computation Workshop. Seattle, Washington, USA. New York: ACM.

BALL, K. 2007: *Conversation about the grid related growth model.*

BENTLEY, P. (ED.) 1999: *Evolutionary Design by Computers.* San Francisco: Morgan Kaufmann.

BURRY, M. 2002: *Rapid prototyping, CAD/CAM and human factors.* In Automation in Construction 11, pp. 313–333.

COATES, P. 2010: *Programming architecture.* New York: Routledge.

COATES, P., BROUGHTON, T., JACKSON, H. 1999: *Exploring 3D design worlds using Lindenmayer systems and genetic programming.* In Peter Bentley (Ed.): Evolutionary design by computers. San Francisco: Morgan Kaufmann, pp. 323–341.

DE LA BARRERA, C.: *Designemergente.* Genr8. Genr8. Available online at <http://www.designemergente.org/experiment/genr8>, accessed 22/05/2013.

DESIGNTOPRODUCTION: *Design to Production.* Available online at <http://www.designtoproduction.com>, accessed 22/05/2013.

DIERICHS, K., FLEISSNER, F., MENGES, A. 2011: *Interrelation of experiment and simulation in the development of Aggregate Architectures.* In Digital Proceedings of the International Symposium on Algorithmic Design for Architecture and Urban Design (ALGODE). Tokyo, Japan.

DOSCHER, M. 2006: *One order on top of the other.* SmartGeometry Conference. Cambridge, UK.

D-SHAPE N.D: *3D printing.* D-shape. Available online at <http://d-shape.com>, accessed 22/05/2013.

EIGENSATZ, M., KILIAN, M., SCHIFTNER, A., MITRA, N. J., POTTMANN, H., PAULY, M. 2010: *Paneling architectural freeform surfaces.* In ACM SIGGRAPH 2010 29 (4).

ERDINE, E. 2008: *Möbius-Klein nonmanifold pavilion.* SmartGeometry Conference. Available online at <http://www.elif-erdine.com/smart-geometry-2008>, accessed 22/05/2013.

EVOLUTE: *Subdivision service.* Available online at <http://www.evolute.at>, accessed 22/05/2013.

FRAZER, J. 1995: *An Evolutionary Architecture.* Architectural Association Publications.

GENERATIVECOMPONENTS: *Bentley Systems.* Available online at <http://www.

bentley.com/en-US/Promo/Generative+Components>, accessed 22/05/2013.

GLYMPH, J., SHELDEN, D., CECCATO, C., MUSSEL, J., SCHOBER, H. 2004: *A parametric strategy for free-form glass structures using quadrilateral planar facets.* In Automation in Construction 13, pp. 187–202.

HEMBERG M. 2001: *GENR8 - A Design Tool for Surface Generation.* MSc Department of Physical Resource Theory. Massachusetts Institute of Technology.

HEMBERG, M., MENGES, A., JONAS, K., GONÇALVES, M. D.C., FUCHS, S. R. 2008: *Genr8: Architects' Experience with an Emergent Design Tool.* In Juan Romero and Penousal Machado (Eds.): The Art of Artificial Evolution. A Handbook on Evolutionary Art and Music; with 30 tables and DVD-ROM. Berlin, Heidelberg, New York: Springer.

HEMBERG, M., O'REILLY, U.M. 2004: *Geometry as a substitute for structural analysis in generative design tools.* In First International Conference on Design Computing and Cognition (DCC'04). Cambridge, Massachusetts.

HILLIS, W. D. 2001, C1998: *The Pattern on the Stone. The Simple Ideas That Make Computers Work.* London: Phoenix.

HUDSON, R. 2010: *Strategies for Parametric Design in Architecture. An application of practice led research.* PhD, Department of Architecture and Civil Engineering, Bath. University of Bath.

HUSBANDS, P., JERMY, G., MCILHAGGA, M., IVES, R. 1996: *Two Applications of Genetic Algorithms to Component Design.* In Terence C. Fogarty (Ed.): Evolutionary Computing. AISB Workshop, Brighton, U.K., April 1-2, 1996: Selected Papers. AISB Workshop. Berlin, Heidelberg, New York: Springer, pp. 50–61.

JANSSEN, P., FRAZER, J., MING-XI, T. 2002: *Evolutionary Design Systems and Generative Processes.* In Applied Intelligence 16 (2), pp. 119–128.

JONAS, K. 2014: *Method for generating a kit of elements for composing or constructing diverse gap-less 3D structures and such kit of elements.* European Patent Application EP14160345.6. 17 March 2014.

JONAS, K., HEMBERG, M. 2006: *ECSS – Evolutionary Component Surface System. A tool for architectural form generation and form optimization.* In GECCO '06, Genetic and Evolutionary Computation Conference. User-centric Evolutionary Computation Workshop. Seattle, Washington, USA. New York: ACM.

LIU, Y., POTTMANN, H., WALLNER, J., YANG, Y.L., WANG, W. 2006: *Geometric modeling with conical meshes and developable surfaces.* In ACM Siggraph 2006.

LUTTON, E., CAYLA, E., CHAPUIS, J. 2003: *ArtiE-Fract: The Artist's Viewpoint.* In Conor Ryan (Ed.): Genetic programming. 6th European Conference, EuroGP 2003, Essex, UK, April 14-16. Proceedings. European Conference on Genetic Programming, EuroGP. Berlin: SpringerLink.

MITCHELL, M. 1998, C1996: *An Introduction to Genetic Algorithms.* 1st MIT Press pbk. Cambridge, Massachusetts: MIT Press.

PEDRESCHI, R. 2000: *Eladio Dieste - The Engineer's Contribution to Contemporary Architecture.* London: Thomas Telford.

POTTMANN, H., ASPERL, A., HOFER, M., KILIAN, A. 2010: *Architekturgeometrie.* Wien, New York: Springer; Bentley Inst. Press.

ROMERO, J., MACHADO, P. 2008: *The Art of Artificial Evolution. A Handbook on Evolutionary Art and Music.* Berlin, Heidelberg, New York: Springer.

SCHEURER, F. 2003: *The Groningen Twister - An experiment in applied generative design.* Available online at <http://wiki.arch.ethz.ch/twiki/pub/Extern/GroningenTwister/GA2003-Scheurer.pdf>, accessed 22/05/2013.

SHEA, K. 2003: *Digital canopy: high-end computation, low-tech construction.* In Architectural Research Quarterly 6 (3), pp. 230–245.

SHELDEN, D.R. 2002: *Digital Surface Representation and the Constructibility of Gehry's Architecture.* PhD, Department of Architecture. Massachusetts Institute of Technology. Cambridge, Massachusetts.

SMARTGEOMETRY. Available online at <http://smartgeometry.org>, accessed 22/05/2013.

SMART: *SMART Form.* Buro Happold SMART. Available online at <http://www.smart-solutions-network.com/page/smart-form>, accessed 22/05/2013.

SMART SOLUTIONS: *SMART Solutions.* Available online at <http://www.burohappold.com/buildings/building-fabric/smart-solutions>, accessed 22/05/2013.

STEADMAN, P. 2008: *The Evolution of Designs.* Biological Analogy in Architecture and the Applied Arts. 2nd: Routledge.

STEPHAN, S., SANCHEZ-ALVAREZ, J., KNEBEL, K. 2003: *Reticulated Structures on Free-Form Surfaces.* Mero. Available online at <http://www.mero.de/fileadmin/downloads/bausysteme/publikationen/free_ret_stru_e.pdf>, accessed 22/05/2013.

TESTA, P., WEISER, D.; O'REILLY, U.M. 1997: *Emergent Design Group (EDG).* Available online at <http://web.mit.edu/edgsrc/www>, accessed 22/05/2013.

VERBUS. *Verbus modular system.* http://www.verbussystems.com/.

VON NEUMANN, J., BURKS, A. W. 1966: *Theory of Self-Reproducing Automata.* Urbana and London: University of Illinois Press.

WHITEHEAD, H.; PETERS, BR. 2008: *Geometry, form and complexity.* In Littlefield, D. (Ed.): Space Craft - Developments in Architectural Computing. London: RIBA.

WINSLOW, P. 2009: *Synthesis and Optimisation of Free-Form Grid Structures.* PhD, Department of Engineering, Cambridge, UK. University of Cambridge.

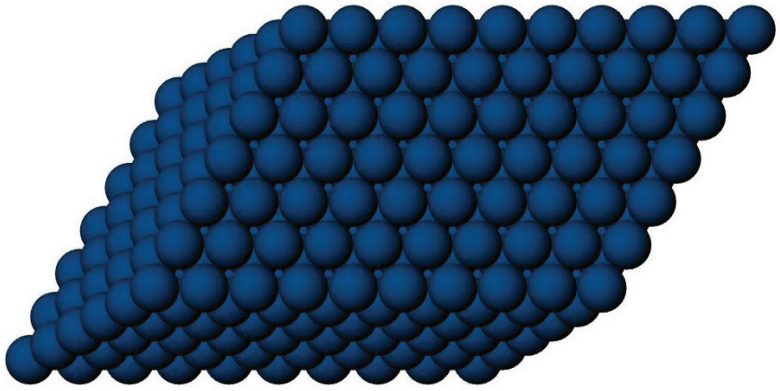
APPENDIX

APPENDIX

IMPLEMENTATION OF THE ‘SURFACES WITHIN A SPHERE NETWORK’ EXPERIMENT

This appendix documents the implementation of the ‘surfaces within a sphere network’ experiment. Two aspects of the experiments are described in more detail than in the main text of this thesis. Firstly the organisation of spheres in the AB_AB_AB configuration and secondly the definition of the triangular surface elements within the lattice that results when connecting the centre points of the spheres.

A. Placing the spheres



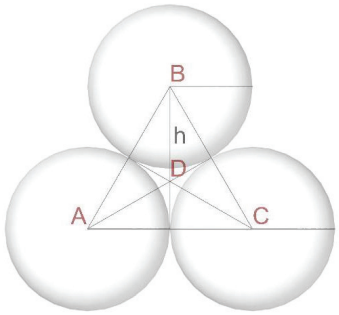
Close sphere packing. Rendering showing a block of spheres packed in a hexagonal ABABAB close packing configuration.

r = Radius

i = incrementing in x

j = incrementing in y

k = incrementing in z



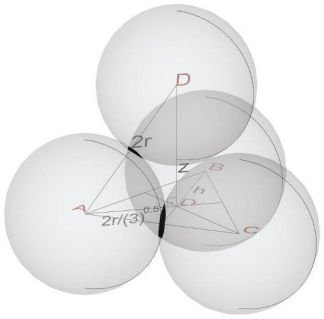
Example for first case, placement for four spheres. Centre points, were D is the first sphere in the next level.

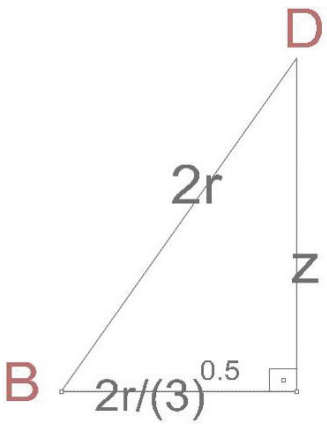
General configuration.

	x	y	z	
A	0	0	05,	
B	r	r√3	0	2D (sphere centre points in first level)
C	2r	0	0	
D	r	r√3	?s	3D (sphere centre point in next level)

How to get to the next level.

	x	y	z
A	i	j	k
B	r	r√3	k
C	2r	j	k
D	r	$\frac{r\sqrt{3}}{3}$	?





$$z^2 + \left(\frac{2r}{r\sqrt{3}}\right) = (2r)^2$$

$$z^2 + \frac{4r^2}{3} = 4r^2$$

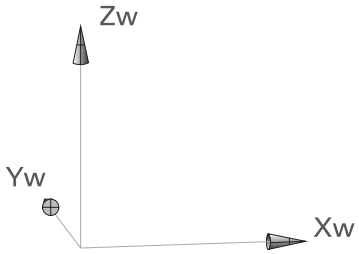
$$z^2 = r^2\left(4 - \frac{4}{3}\right)$$

$$= r^2\left(\frac{12}{3} - \frac{4}{3}\right)$$

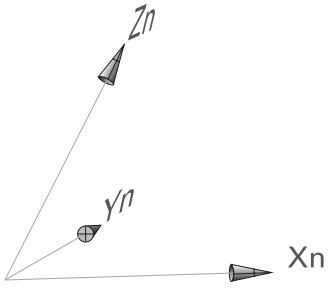
$$= r^2\left(\frac{8}{3}\right)$$

$$z = \sqrt{\frac{8}{3}} * r$$

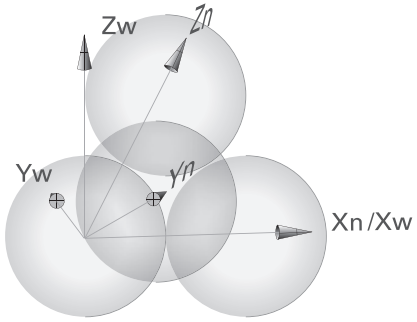
The next step is to propagate the spheres into space along a vector. To do so in an automated parametric way, the world coordinate system needs to be transformed into a coordinate system in which all vectors are shifted by a value. In our case the shift has a linear relation to the increments of i, j and k. The parameters are determined by trigonometrically deriving the position change.



World coordinate system.



New coordinate system (index coordinate system for sphere).



World and index coordinate system within a configuration of three spheres.

A.1. Defining a difference-operator, to determine the change in the position value for each sphere centre point

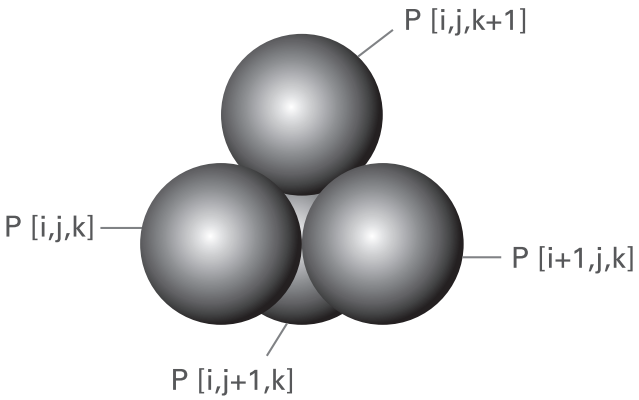
Δ = change in position value

P = centre point

i = index along the row

j = index walong column

k = index along level



The current sphere centre point index is subtracted from the next sphere centre point index in order to find the change in the position value.

$\Delta P_i = P[i+1,j,k] - P[i,j,k]$

$\Delta P_j = P[i+1,j,k] - P[i,j,k]$

$\Delta P_k = P[i+1,j,k] - P[i,j,k]$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}_i * i + \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix}_j * j + \ldots$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (\Delta x_i i + \Delta y_j j + \Delta x_k k) \\ (\Delta y_i i + \Delta y_j j + \Delta y_k k) \\ (\Delta z_i i + \Delta z_j j + \Delta z_k k) \end{bmatrix}$$

Δx_i = the change in the x-coordinate created by a change in the i-coordinate, e.g. if i increases by 2 then $\Delta x = \Delta x_i * 2$

$$\begin{matrix} d_x \\ d_y \\ d_z \end{matrix} = \begin{bmatrix} \frac{\partial x}{\partial i} & \frac{\partial x}{\partial y} & \frac{\partial x}{\partial k} \\ \frac{\partial y}{\partial i} & \frac{\partial y}{\partial y} & \frac{\partial y}{\partial k} \\ \frac{\partial z}{\partial i} & \frac{\partial z}{\partial y} & \frac{\partial z}{\partial k} \end{bmatrix} \begin{bmatrix} di \\ dj \\ dk \end{bmatrix}$$

$$\begin{matrix} x \\ y \\ z \end{matrix} = r \begin{bmatrix} 2 & 1 & 1 \\ 0 & \sqrt{3} & \frac{1}{\sqrt{3}} \\ 0 & 0 & \sqrt{\frac{8}{3}} \end{bmatrix} \begin{bmatrix} i \\ j \\ k \end{bmatrix}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = r \begin{bmatrix} 2i \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} j \\ \sqrt{3}j \\ 0 \end{bmatrix} + \begin{bmatrix} k \\ \frac{k}{\sqrt{3}} \\ \sqrt{\frac{8}{3}}\,k \end{bmatrix}$$

This results in the following formulas for placing spheres in a crystal formation, closest packed.

$$x = r(2i + j + k)$$

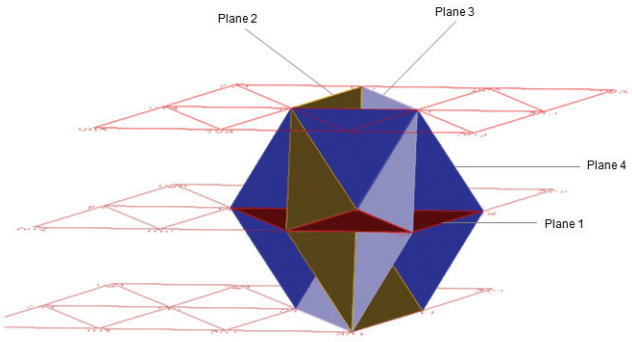
$$y = r(\sqrt{3}j + \frac{k}{\sqrt{3}}$$

$$z = r\sqrt{\frac{8}{3}}\,k$$

Note: the x-axis is aligned with the i-axis.

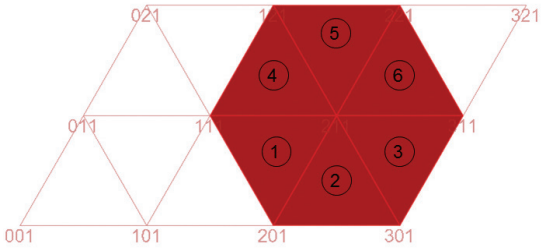
B. Defining the triangular surface elements

The following documentation shows how a list of neighbouring triangles to each point in the centre point lattice was defined.



Example showing all triangles that are defined around the sphere centre point 212.

B.1. Plane 1 Triangle neighbours, assigned anticlockwise



Plan view of plane 1.

① 211; 111; 201
i, j, k; i-1, j, k; i, j-1, k

② 211; 201; 301
i, j, k; i, j-1, k; i+1, j-1, k

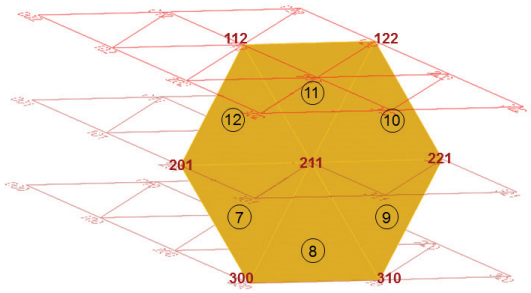
③ 211, 301, 311
i, j, k; i+1, j-1, k; i+1, j, k

④ 211, 121, 111
i, j, k; i-1, j+1, k; i-1, j, k;

⑤ 211, 221, 121
i, j, k; i, j+1,k ; i-1, j+1, k;

⑥ 211, 311, 221
i, j, k; i+1, j, k; i, j+1,k

B.2. Plane 2 Triangle neighbours



Isometric view of plane 2.

⑦ 211; 201; 300
i, j, k; i, j-1, k; i+1, j-1, k-1

⑧ 211; 300; 310;
i, j, k; i+1, j-1, k-1; i+1, j, k-1

⑨ 211; 310; 221
i, j, k; i+1, j, k-1; i, j+1, k

⑩ 211; 221; 122

i, j, k; i, j+1, k; i-1, j+1, k+1

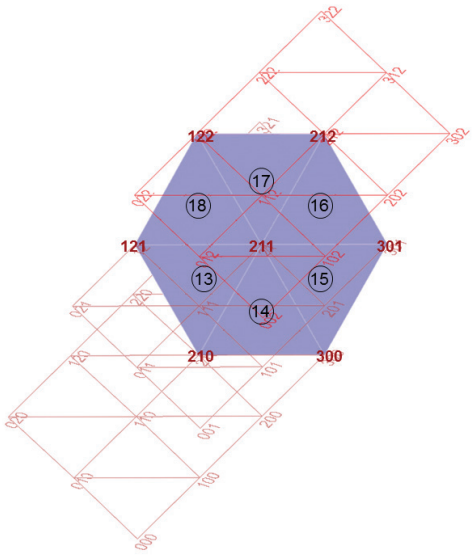
⑪ 211; 122; 112

i, j, k; i-1, j+1, k+1; i-1, j, k+1

⑫ 211; 112; 201

i, j, k; i-1, j, k+1; i, j-1, k

B.3. Plane 3Triangle neighbours



⑬ 211; 121; 210

i, j, k; i-1, j+1, k; i, j, k-1

⑭ 211; 210; 300

i, j, k; i, j, k-1; i+1, j-1, k-1

⑮ 211; 300; 301

i, j, k; i+1, j-1, k-1; i+1, j-1, k

⑯ 211; 301; 212

i, j, k; i+1, j-1, k; i, j, k+1

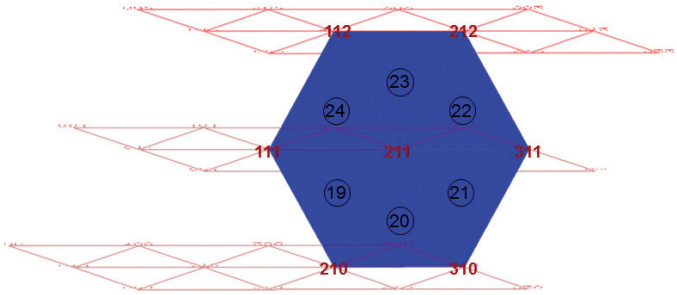
⑰ 211; 212; 122

i, j, k; i, j, k+1; i-1, j+1,k+1

⑱ 211, 122, 121

i, j, k; i-1, j+1,k+1 i-1, j+1, k

B.4. Plane 4 Triangle neighbours



⑲ 211; 111; 210

i, j, k; i-1,j, k; i, j, k-1

⑳ 211; 210; 310

i, j, k; i, j, k-1; i-1,j, k;

㉑ 211; 310; 311

i, j, k; i+1, j, k-1; i+1, j, k

㉒ 211; 311; 212

i, j, k; i+1, j, k; i, j, k+1

㉓ 211; 212; 112

i, j, k; i, j, k+1; i-1, j, k+1

㉔ 211; 112; 111

i, j, k; i-1, j, k+1; i-1, j, k